CS440: Introduction to Software Engineering

Requirements Specification

Group members:
Kamran Lodhi mlodhi3@uic.edu
Nazari Skrupsky nskrou2@uic.edu
Nestor Sotres nsotre2@uic.edu
Michal Szumilo mszumi2@uic.edu

1. Product Overview and Summary

1.1 Product Overview

Business Model

This particular university runs 2 semesters per year, a Spring and Fall Semester. At the beginning of each semester, classes to be offered are scheduled, assigned Professors, Teaching Assistants, time slots, and locations. After the classes have been prepared, students are allowed to search for, register, and add/drop classes. There will be a gracing period of two weeks to drop a class without a penalty. After half a semester each student will be required to see counselor who will remove advising hold.

Students will also need to review their graduation audit in order to see what classes have been taken and which are still needed. G.P.A. is based on a 4.0 grading scale. After students have met requirements, they are able to apply for graduation.

1.2 Product Summary

The software will facilitate enrollment to classes, reviewing/editing schedules, review/apply for graduation for students by providing an easy to use and intuitive interface.

Professors, Teachers Assistants (TA), and Advisers will also benefit from the user friendly interface by allowing them to quickly view class schedules, list of students, viewing/editing a calendar of events, and perform various administrative procedures.

In order to manage such a large system an Administrative (Admin) roll has been created in order to manage several different settings, database processes, users and events.

2. Development, Operational and Maintenance Environment

The system will be developed using C++ language along with relevant UI classes. The mysql server or Microsoft SQL server will be used for database tables. The design phase will yield further information on which database program will be used.

The system will be deployed on Windows, Mac OS, and Linux operating systems.

3. Use Cases and their complete description

3.1 Actors and Initial Requirements

The initial requirements were taken from a series of meetings describing what each user (actor) must be able to do within the application. These actions have been compiled in list form for legibility purposes.

3.1.1 Student

- Modify Personal Information (certain fields on profile)
- Look up classes offered this semester
- Register for a class
- Drop a class
- View overall class schedule (classes signed up for)
- See available times for day/week and show available classes (sort of a schedule maker)
- Contact Professor
- See classes that are full
- Get on a waiting list for a class
- View class detail (course web page, instructor, TA, Date/Time, location)
- View Graduation Audit
- Apply for Graduation
- Receive email reminder about add/drop deadlines
- Print out schedule of classes
- View Status (full time/ part time)
- View holds

3.1.2 Professor

- Modify personal information
    - modify name/password/contact information
- able to see the courses that he teaches during the semester
- able to see courses that he has taught previously
- able to see list of students taking his course
- message manager

    - send notifications to users. compose and send messages to users or groups of users
    - read received messages
- contact TA (if assigned)

3.1.3 Teacher Assistant

- Modify personal information
    - modify name/password/contact information
- able to see the courses that he assisting in during the semester
- able to see his previous assignments as Teacher Assistant
- able to see list of students taking his assisted course
- message manager
    - send notifications to users. compose and send messages to users or groups of users
    - read received messages

3.1.4 Adviser/ Counselor

- requirements inherited from teacher
- login to his/hers account
- pick a student to be his adviser
- change student name
- view student's class schedule
- view student's progress (GPA/courses taken)
- override class capacity
- create appointment
- modify appointment
- check graduation requirements
- clear advising holds
- approve self registration

3.1.5 Administrator

- manage user account
    - add/remove/modify user: name, surname, address, phone, email, emergency contact, id (automatically generated), role (student, teacher, ta, advisor, administrator)

    - add/remove types of holds: immunization, advising, unpaid tuition, academic standing

- manage subject
    - add or modify subjects, e.g., computer science, electrical and computer engineering, architecture, etc

- manage course
    - add/remove/modify courses: description, prerequisite requirements, number of credit hours

- manage term
    - add/remove/modify terms, e.g., fall 2011, spring 2012. subjects and courses can be added and removed to terms.
    - add and remove subjects for each term
    - add/remove/modify courses to subjects for each term: classroom, meeting time, class capacity, instructor, ta
    - add/remove term registration period

- import application data
    - batch import users, subjects, and courses from file

- manage calendar
    - add/remove/modify events to calendar

- message manager
    - send notifications to users. compose and send messages to users or groups of users
    - read received messages

3.2 Use Case Diagrams for Actors

3.2.1 Use Case Diagrams for Student

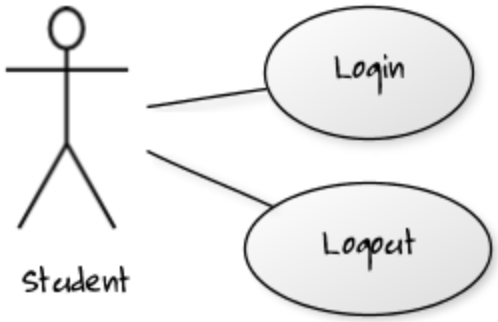


figure 1: Login Logout

Brief Description:

The ***Login*** use case allows the Student to log into the application.

Step-by-Step Description:

1. Student will open the executable file for the application
2. Student will enter their user name and password to login

Brief Description:

The ***Logout*** use case allows the Student to log out of the application.

Step-by-Step Description:

1. When the Student is done using the application, he/she will click on the logout option to log out of the application

---

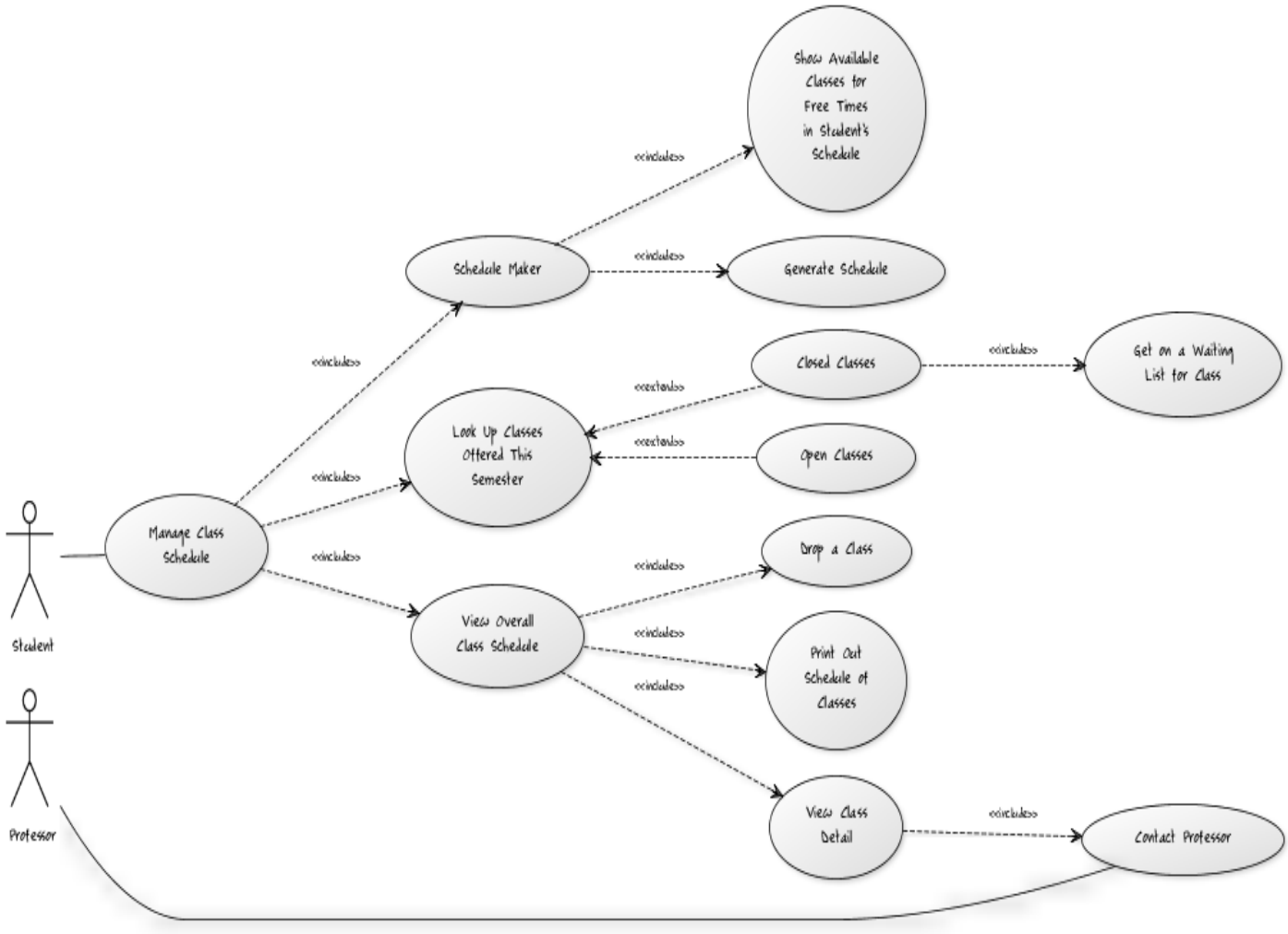


figure 2: Manage Class Schedule

Brief Description:

The ***Manage Class Schedule*** will allow the student to perform several actions regarding their scheduling and registration of classes.

Step-by-Step Description:

1. Student can get automated help in creating a schedule by using the *Schedule Maker* use case.

    1.1 Student can have the automated system display a list of classes that can fill their specified schedule parameters using the *Show Available Classes for*

Free Times in Student's Schedule use case.

1.2 Student can have the automated system create a schedule for him/her using the *Generate Schedule* use case.

2. Student can search for classes offered in the current semester using the *Look Up Classes Offered This Semester* use case.

2.1 Student can view closed classes by using the *Closed Classes* use case.

2.2 Student can sign up for a waiting list using the *Get on a Waiting List for Class* use case.

2.2 Student can view classes still available for enrollment using the *Open Classes* use case.

3. Student can view their current semester's enrollment schedule using the *View Overall Schedule* use case.

3.1 Student can drop a class using the *Drop a Class* use case.

3.2 Student can print their current class schedule using the *Print Out Schedule of Classes* use case.

3.3 Student can view the details of an enrolled class using the *View Class Detail* use case.

3.3.1 Student can send an email to the Professor using the *Contact Professor* use case.

Brief Description:

The *Schedule Maker* use case will allow a Student to create a class schedule by automatically generating a schedule or listing classes that will fit the Student's open time slots.

Step-by-Step Description:

1. The Student can choose to have the schedule maker show what classes are available for open time slots in the Student's schedule.

1.1 The schedule maker will take the Student's ID and up to 5 course ID's the student is interested in taking as input and generate a list of classes offered within open times in the Student's schedule.

2. The Student can opt to have the schedule maker generate an entire schedule for them.
    1.1 The schedule maker will take up to 5 course ID's the student is interested in taking as input and generate up to 5 possible schedules for the student showing the different times and classes available for each schedule in a visual report.

Brief Description:

The ***Look Up Classes Offered This Semester*** will allow a Student to query the course catalog for classes offered during the current semester and view open and closed status. In the case of a class that is closed, the Student will be allowed to get on a waiting list for enrollment.

Step-by-Step Description:

1. Student can search for all open classes or narrow a search by the following criteria:
    1.1 Criteria are as follows:
        - Class name
        - Department (Math, English, Computer Science, Art, etc.)
        - Days offered
        - Time
        - Open registration
        - Closed registration
    1.2 If the class searched for is closed for registration the Student can get on a registration waiting list.

Brief Description:

The ***View Overall Class Schedule*** will allow the Student to view, edit, and print their current schedule. It will also allow them to contact the Professor for the class they're enrolled for.

Step-by-Step Description:

1. Student can view a generated graphical representation of the Students schedule.
    1.1 The graphical representation can be in the following format:
        - Detail view (list)
            - Student can send the Professor a message while in Detail view using the *Message Manager* use case.
        - Weekly view (calendar view)
    1.2 The following information will be displayed
        - Class name
        - Class time
        - Days class is offered
        - Building and room number for class
        - Professor
        - TA if any
        - Credit hours for class (detail view)
        - Link to contact professor

2. Student can edit their schedule by dropping any of the classes they're enrolled for.

3. Student can print their schedule - generated on demand.
    3.1 Student can select printing the Detailed or Weekly view schedule.

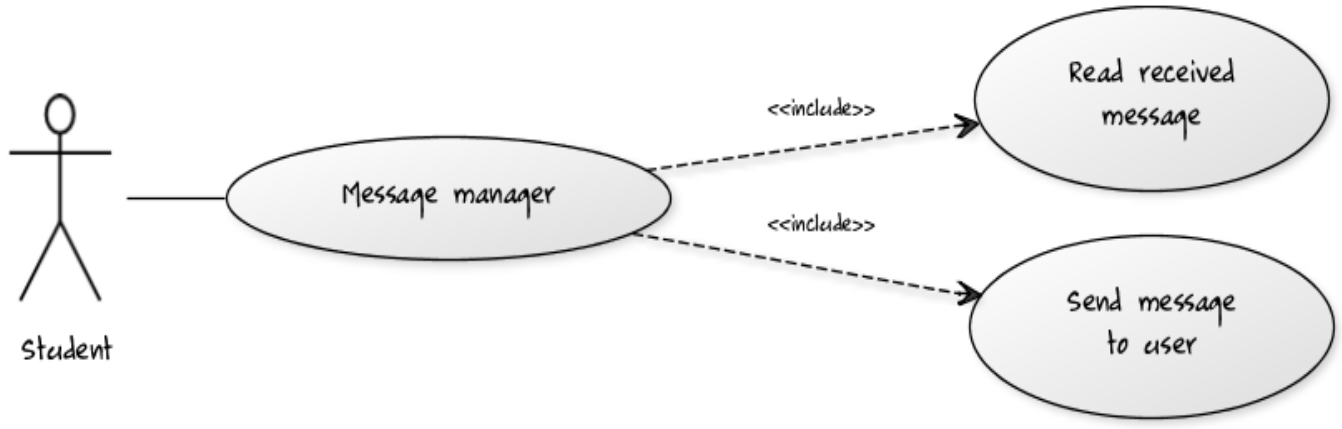


figure 3: Message Manager

Brief Description:
The *message manager* case allows an Student to send messages to users and read incoming messages.

Step-by-Step Description:
1. Compose and send a message to a user or read received messages.

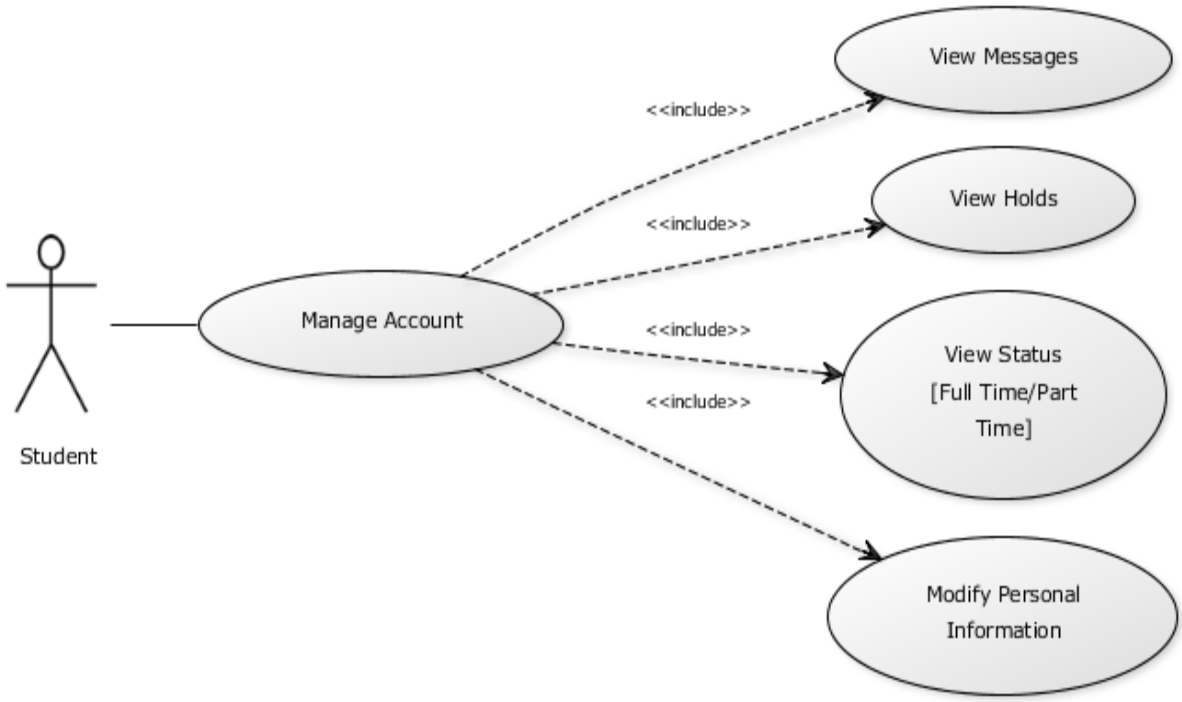


figure 4: Manage Account

Brief Description:

The *Manage Account* use case will allow a Student to view and edit several different things on their student account.

Step-by-Step Description:

1. Student can view their reminders using the *View Message* use case.

2. Student can view holds on their account using the *View Holds* use case.

3. Student can view their enrollment status using the *View Status [Full Time/Part Time]* use case.

4. Student can edit their account information using the *Modify Personal Information* use case.

Brief Description:

The *View Message* use case will allow the student to view several different university graduation/enrollment reminders that have been sent to their account.

Step-by-Step Description:

1. A list of messages for the student to view must be compiled from newest to oldest by default.
    1.1 Student can sort the messages by:
        - Sender (alphabetically)
        - Importance
        - Date
    1.2 Student can edit the list in the following manner:
        - Delete the reminder
        - Mark as important

Brief Description:

The *View Holds* use case allows a student to see any holds on his/her student account

Step-by-Step Description:

1. A list of holds will be generated for the student to view

1.1 The holds will display the following information

- Type of hold
- Department
- Detail

Brief Description:

The *View Status [Full Time/Part Time]* will allow the student to view their enrollment status.

Step-by-Step Description:

1. A simple listing will show the Student the following information:
    - Enrollment Status
        - Full Time/Part Time
        - Credit hours enrolled for
        - Semester enrolled in

Brief Description:

The *Modify Personal Information* use case will allow the Student to edit information on their student account.

Step-by-Step Description:

1. Student will be displayed their current information:
    1.1 The information to be shown is:
        - Name
        - Address
        - Billing Address
        - Degree/Major
        - Emergency Contact
        - Phone (Cell, Home, Work)
        - email

2. Student will be able to modify the displayed information by clicking “edit”

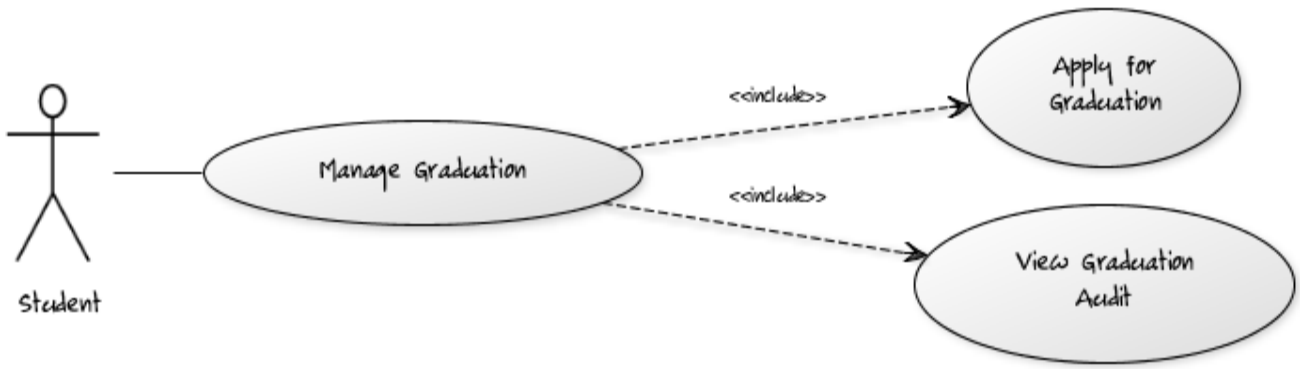


figure 5: Manage Graduation

Brief Description:

The ***Manage Graduation*** use case allows a student to view their graduation audit report and apply for graduation once their degree requirements are met.

Step-by-Step Description:

1. Student can view their graduation audit report using the *View Graduation Audit* use case.

2. Student can submit an application for graduation using the *Apply for Graduation* use case.

Brief Description:

The ***View Graduation Audit*** use case allows a student to view a report on their completion progress towards earning their degree.

Step-by-Step Description:

1. A graduation audit report must be generated for the student- printed on demand:

1.1 For each report the following attributes are printed:

- Student name
- Student ID
- Student's Major and Degree
- Students GPA
- Graduation GPA and credit hour requirements
- A list of all required courses that are completed
- A list of all required courses that are not completed
- A list of electives that are not completed
- A list of possible courses to take to fill requirements

Brief Description:

The *Apply for Graduation* use case allows a student to submit an application for graduation to the University

Step-by-Step Description:

1. A set of requirements must be met before this option is available
    - GPA must meet required level
    - All required and elective courses must be completed with an acceptable grade
    - The Student must not have any holds on his/her account
2. A request for graduation is sent to the appropriate department including the following report:
    - Graduation Audit report
    - A list of requirements met
    - A list of requirements not met
    - A list of any holds the student has
    - Date for graduation

3.2.2 Use Case Diagram for Administrator

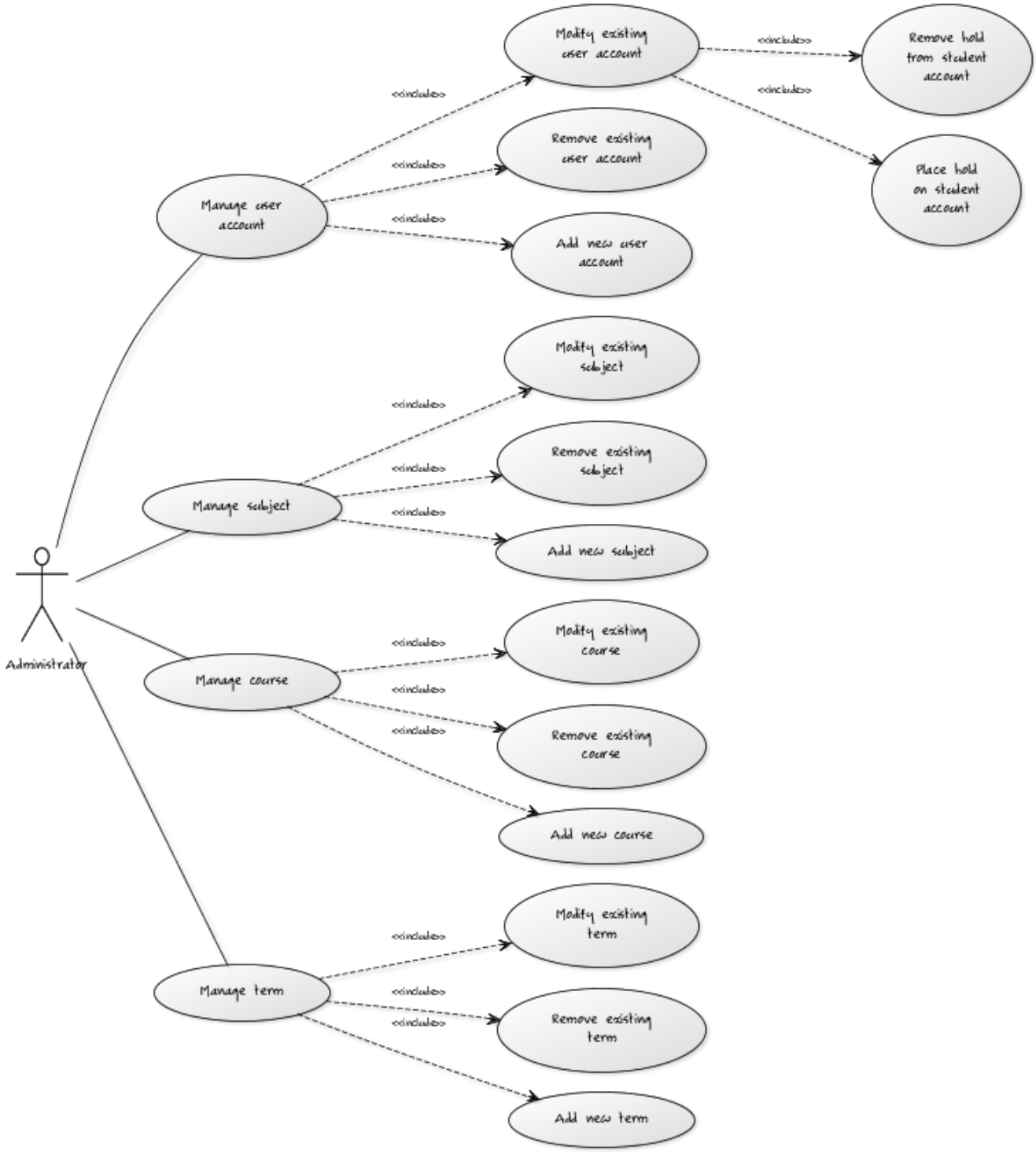


figure 6: Part 1 of Combined Use Case for Administrator

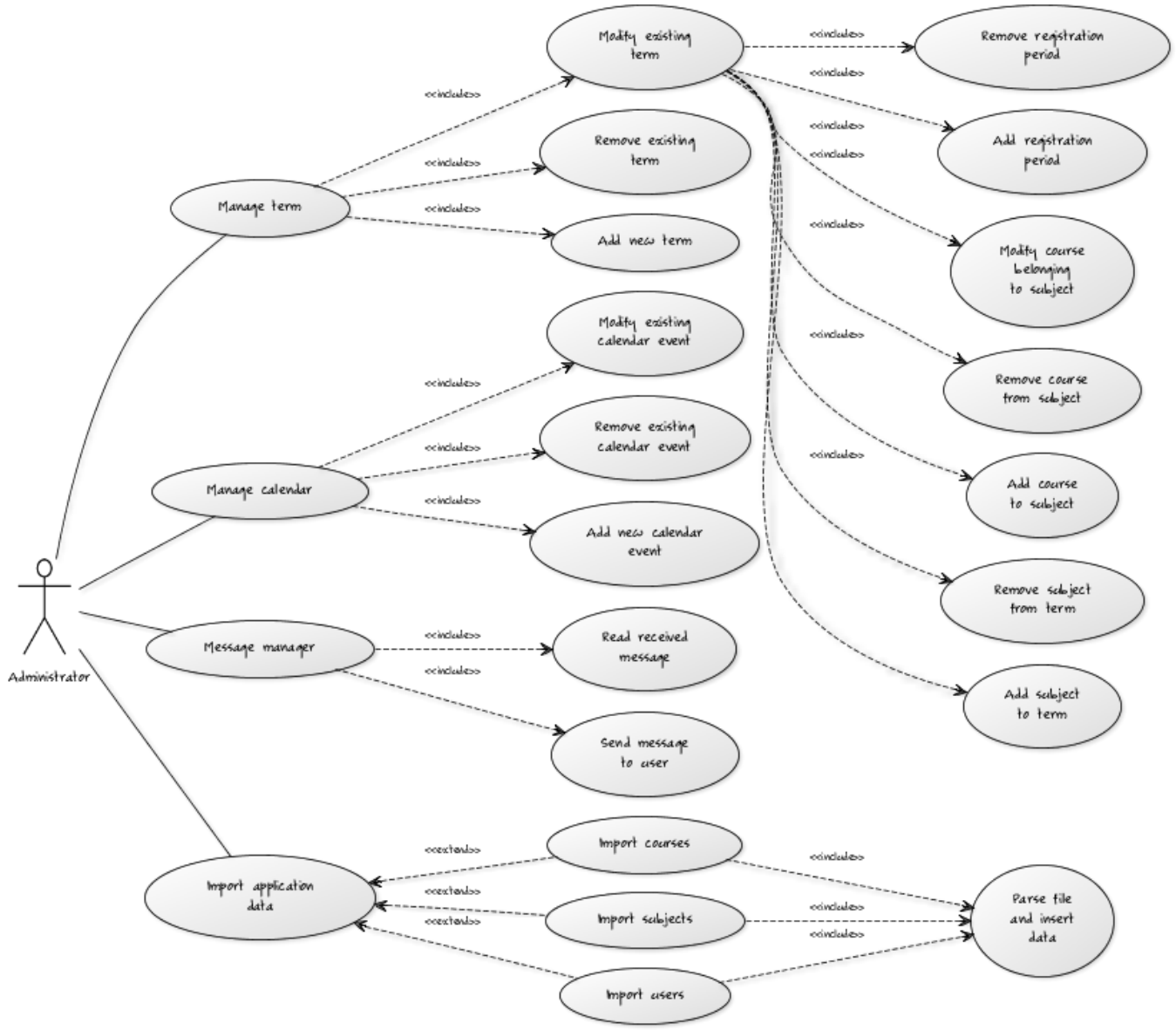
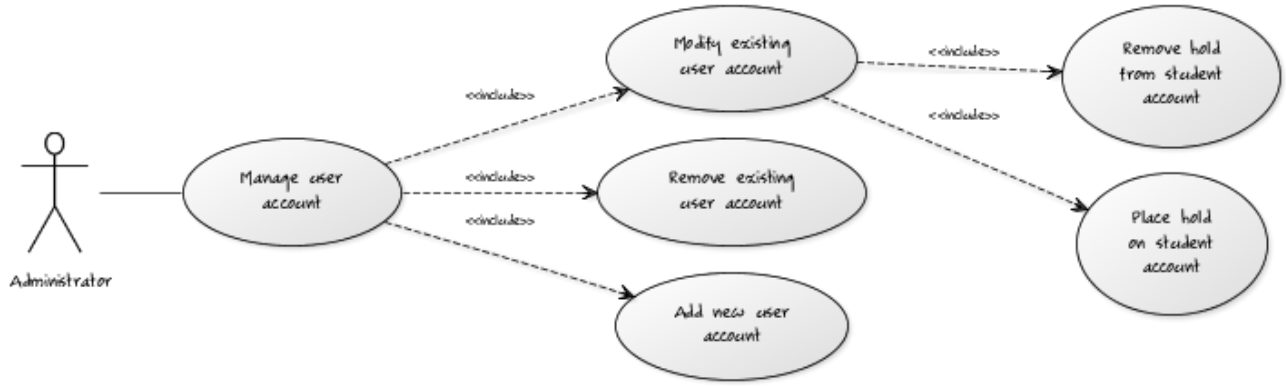


figure 7: Part 2 of Combined Use Case for Administrator

Individual use case diagrams for Administrative role:

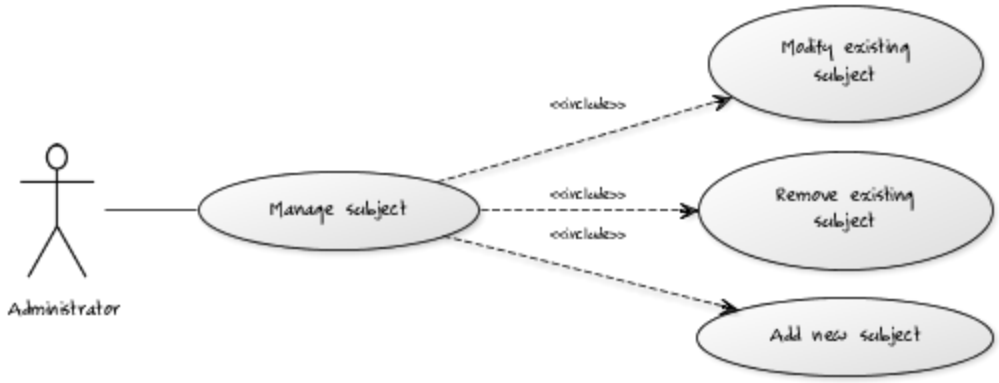


Brief Description:

The *manage user account* case allows an administrator to add and delete users and modify user accounts. Additionally, this case allows adding and removing holds on student accounts. All user accounts consists of name, surname, address, phone, email, id (automatically generated), role (student, teacher, ta, advisor, administrator). Student accounts also contain emergency contact and holds (if any, e.g., immunization, advising, unpaid tuition, academic standing).

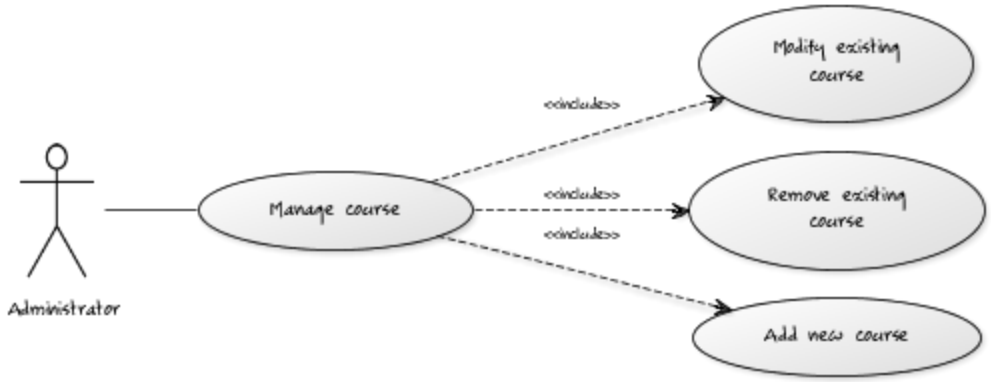
Step-by-Step Description:

1. add, modify, or delete a user account.
2. modifying existing student user account also includes placing and removing holds in addition to standard user account information.



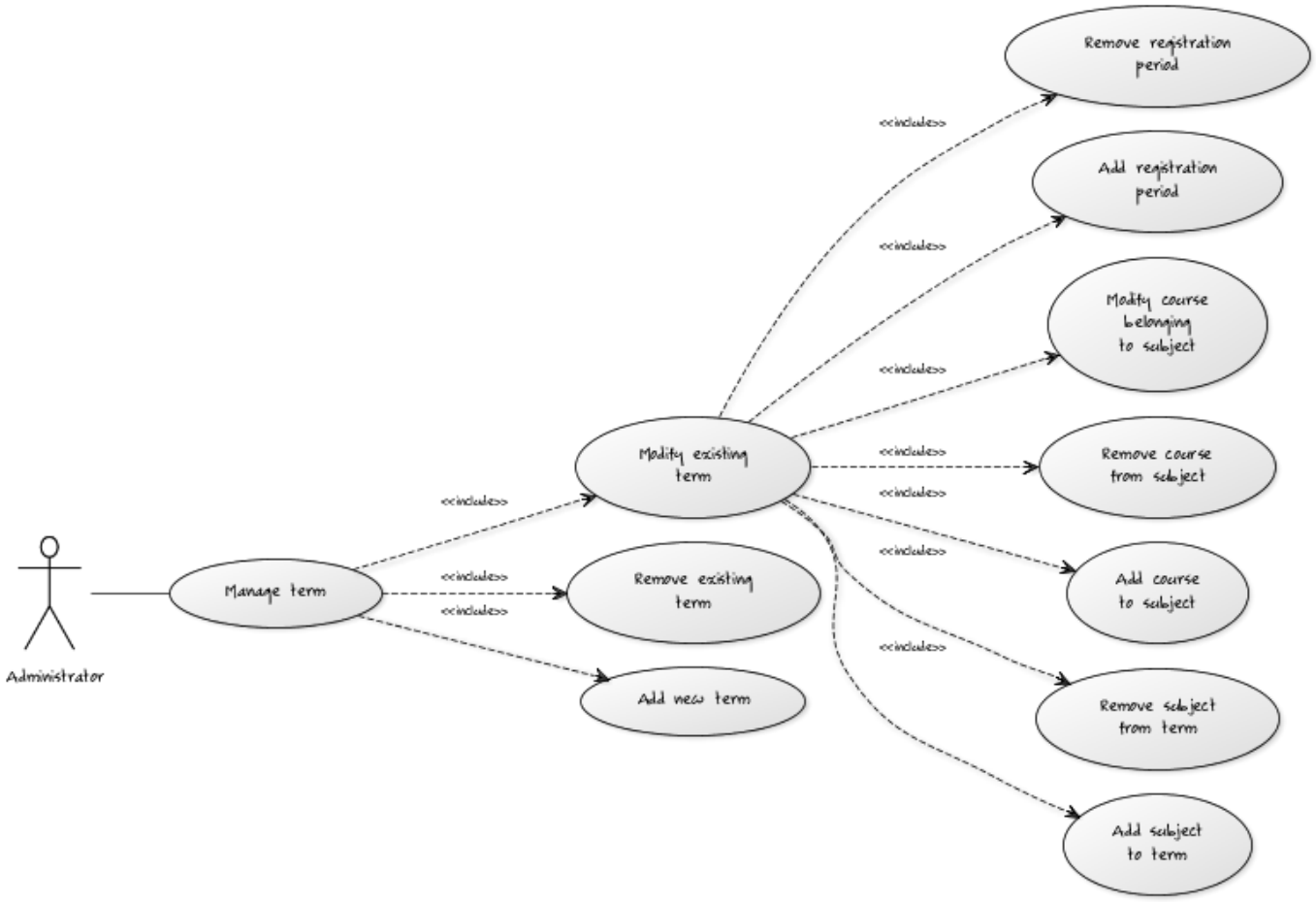
Brief Description:
The *manage subject* case allows an administrator to add and delete subjects and modify subjects. Each subject has a name (e.g., computer science, electrical and computer engineering, architecture, etc.).

Step-by-Step Description:
1. add, delete, or modify a subject.



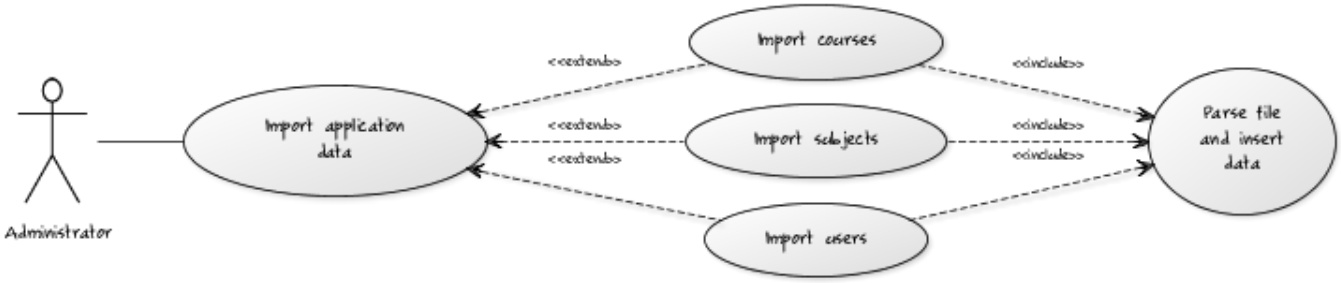
Brief Description:
The *manage course* case allows an administrator to add and delete courses and modify course details. Each course stores constant information (i.e., details about the course that go unchanged between terms) such as description, prerequisite requirements, number of credit hours, etc.

Step-by-Step Description:
1. add, modify, or delete a course.



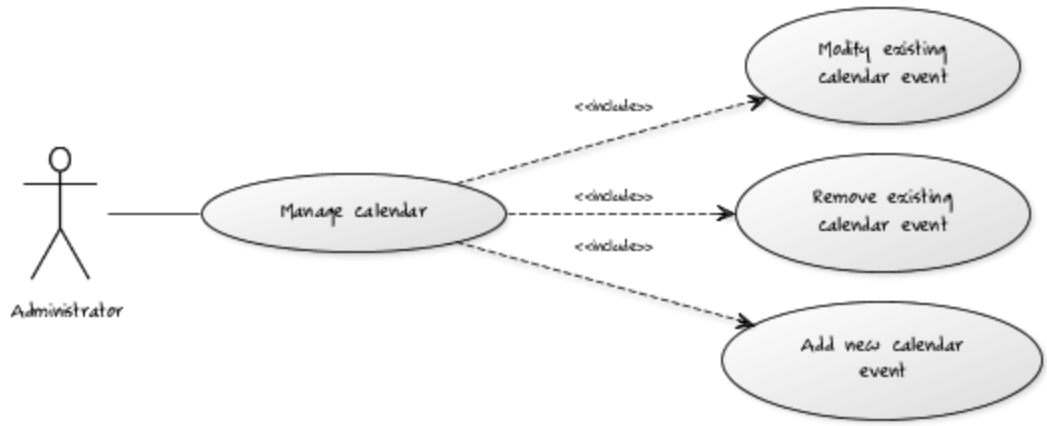
Brief Description:
The *manage term* case allows an administrator to add and delete terms and modify term details, which includes adding and removing subjects for each term; adding and removing courses to subjects as well as modifying course details; and adding and removing term registration period. Each course contains information such as classroom location, meeting time, class capacity, instructor, ta, etc.

Step-by-Step Description:
1. add, delete, or modify a term.
2. modifying a term involves adding and removing subjects; adding and removing courses into subjects and modifying course details; and setting and removing the term registration period.



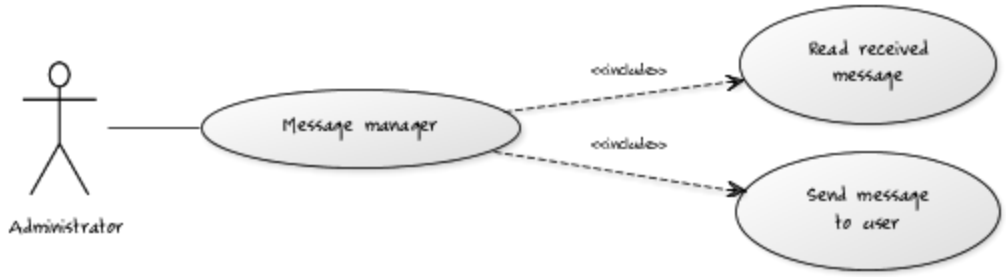
Brief Description:
The *import application data* case allows an administrator to batch import users, subjects and courses.

Step-by-Step Description:
1. import application data.
    1.1. import users.
    1.2. import subjects.
    1.3. import courses.
2. the import process parses the provided file and inserts the data into the database.



Brief Description:
The *manage calendar* case allows an administrator to add, modify, and delete events from a calendar.

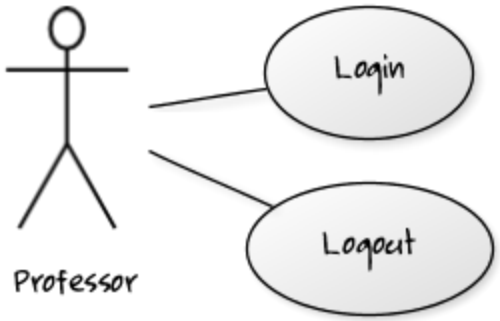
Step-by-Step Description:
1. add, modify, or delete an event from calendar



Brief Description:
The *Message Manager* case allows an administrator to send messages to users and read incoming messages.

Step-by-Step Description:
1. compose and send a message to a user or group of users or read received messages.

3.2.3 Use Cases for Professor



Brief Description:

The *Login* use case allows the Professor to log into the application.

Step-by-Step Description:

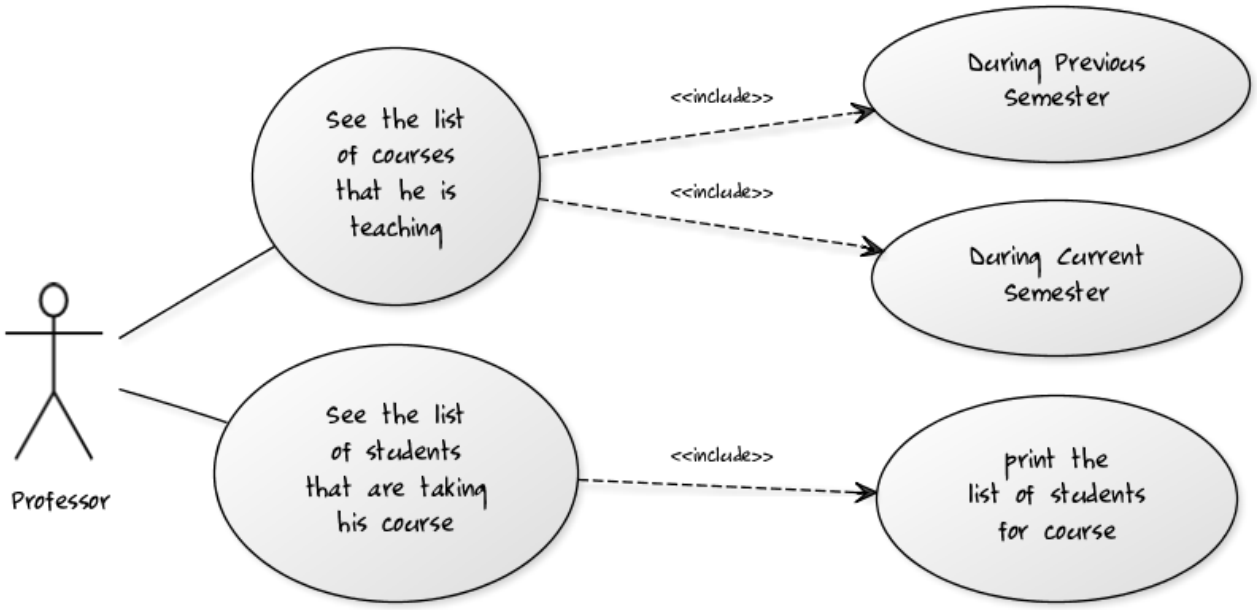
1. The Professor will open the application with the login page at display.
2. He will put in proper credentials to log in the system.

Brief Description:

The ***Logout*** use case allows the Professor to log out of the application.

Step-by-Step Description:

1. When the Professor is done using the application, he/she will click on the logout option to log out of the application.



Brief Description:
The professor will be allowed to see the list of courses that he has taught previously or is teaching them during the current semester.

Step by Step Description:

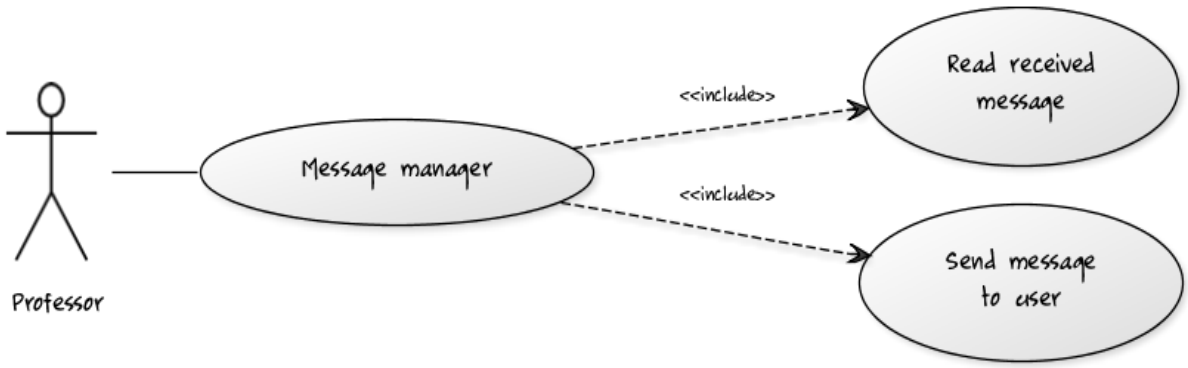
1. After login, the professor will be seeing all the courses that he has taught, either

during the current semester or he has taught previously.

Brief Description:
The professor will be able to see and print the list of students that are taking his course(s) during the current semester.

Step by Step Description:

1. After logging into the system, the professor will click on the course for which he wants to see the list of students
2. After clicking on the course, he will get a list of all the students taking his course.
3. He can then request the system to print the list taking that course.

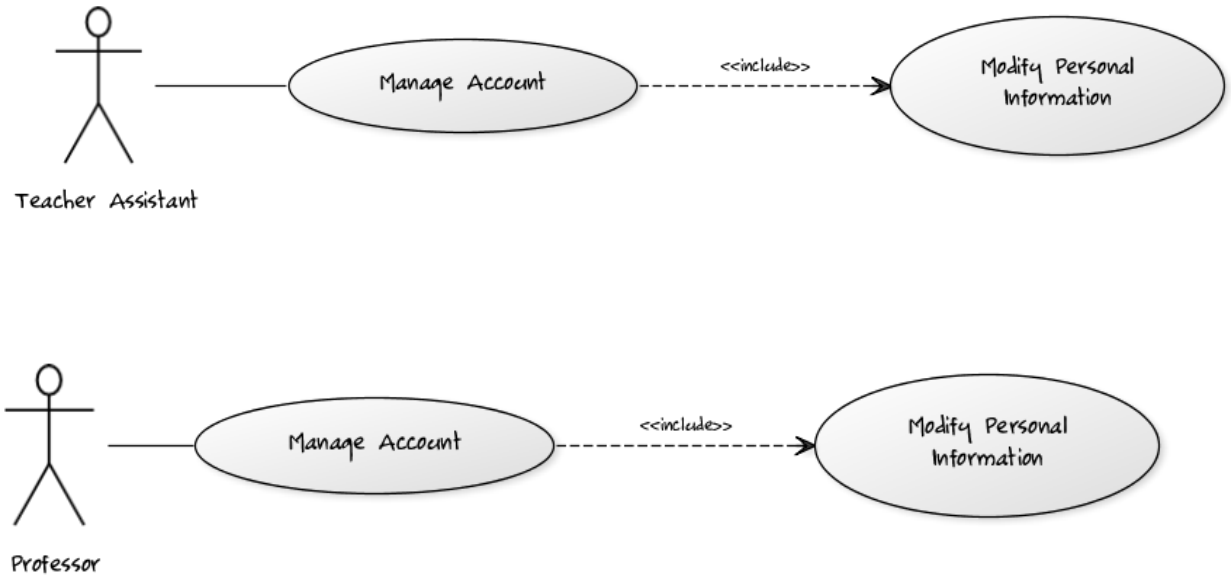


Brief Description:
The professor will be able to send emails to his students or his TA(if assigned)

Step by Step Description:
1. The professor want to send a message.
1.1 The professor wants to send a message to his students.
1.1.1 He will press the link / button for the course to who's students he wants to send the message.
1.2.1 After going to the link, he will send a mass email to the students.

1.2 The professor wants to send a message to his TA
1.2.1 He will press the link/button for the course of whos TA he wants to contact.
1.2.2 After going to the link, he will send a message to TA.

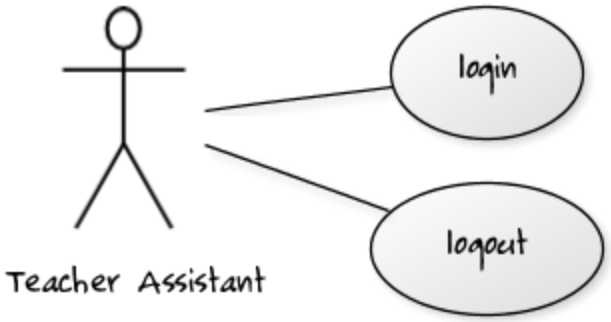
2.The professor wants to read his messages



Brief Description:
The professor / TA will be able to modify his personal information.

Step by step Description:
1. He will be press a button to modify his personal information.
2. He will modify the information that he wants to.
3. After making all the changes, he will press the save button.

3.2.4 Use Cases of Teacher Assistant



Brief Description:

The *Login* use case allows the Professor to log into the application.

Step-by-Step Description:

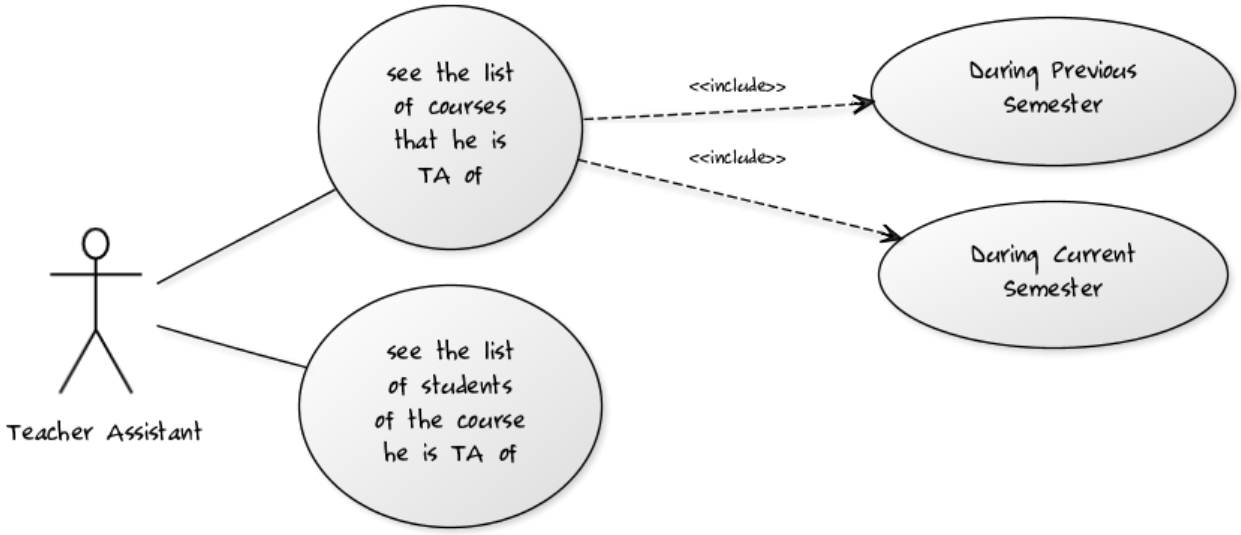
1. The Professor will open the application with the login page at display.
2. He will put in proper credentials to log in the system.

Brief Description:

The *Logout* use case allows the Teacher Assistant to log out of the application.

Step-by-Step Description:

1. When the TA is done using the application, he/she will click on the logout option to log out of the application.



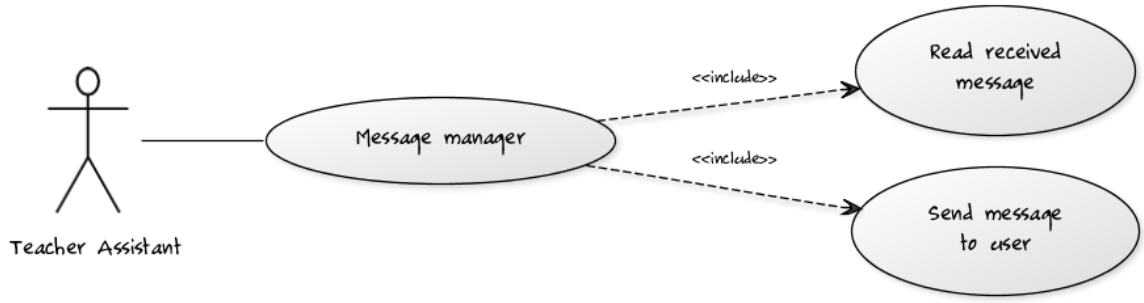
Brief Description:
The TA will be allowed to see the list of courses that he has assisted previously or during the current semester.

Step by Step Description:

1. The teacher assistant will login into the system using his credentials
2. After login, the teacher assistant will be seeing all the courses that he has taught, either during the current semester or he has taught previously.

Brief Description:
The teacher assistant will be able to see the list of students that are taking his course(s), he is TA of, during the current semester.

Step by Step Description:
1. After logging into the system, the TA will be able to look at the list of students that are taking his courses.



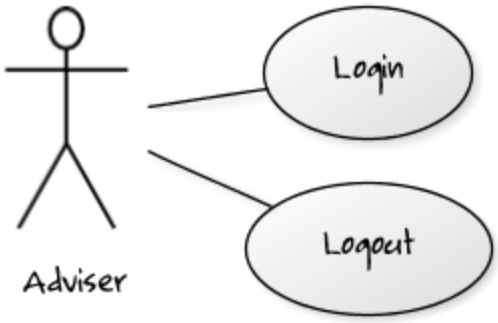
Brief Description:
The Teacher Assistant will be able to send emails to his students or his course professor.

Step by Step Description:
1. The TA want to send a message.
1.1 The TA wants to send a message to his students.
1.1.1 He will press the link / button for the course to who's students he wants to send the message.
1.2.1 After going to the link, he will send a mass email to the students.

1.2 The TA wants to send a message to his professor
1.2.1 He will press the link/button for the course of the professor he wants to contact.
1.2.2 After going to the link, he will send a message to professor

2.The Teacher Assistant wants to read his messages

3.2.5 Use Case Diagrams for Adviser



Brief Description:

The ***Authentication*** use case allows the Adviser to log into the application.

Step-by-Step Description:

1. Adviser will open the executable file for the application
2. Adviser will enter their user name and password to login

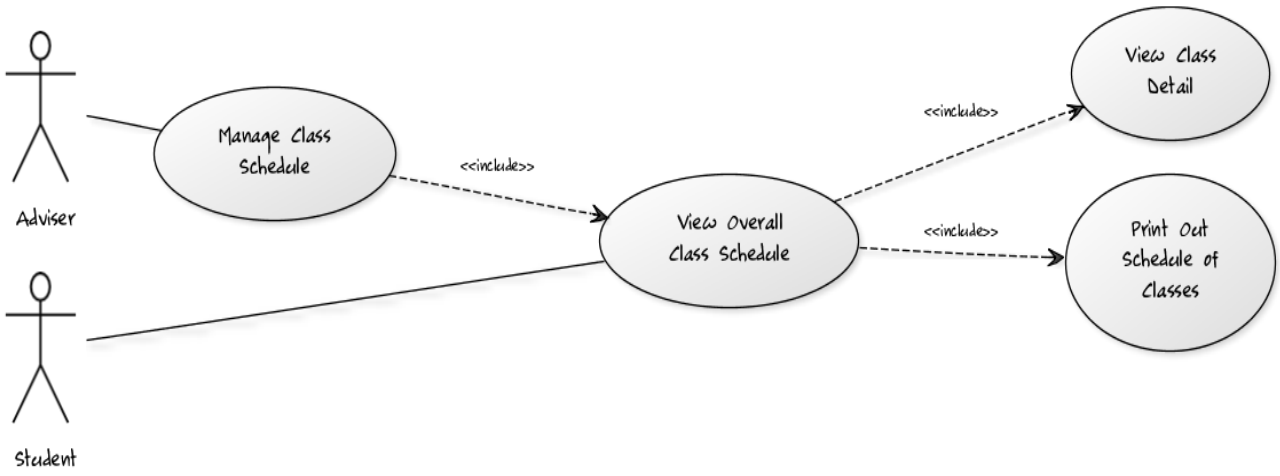
Brief Description:

The ***Logout*** use case allows the Adviser to logout of the application.

Step-by-Step Description:

1. When the Adviser is done using the application, he/she will click on the logout option to log out of the application

---

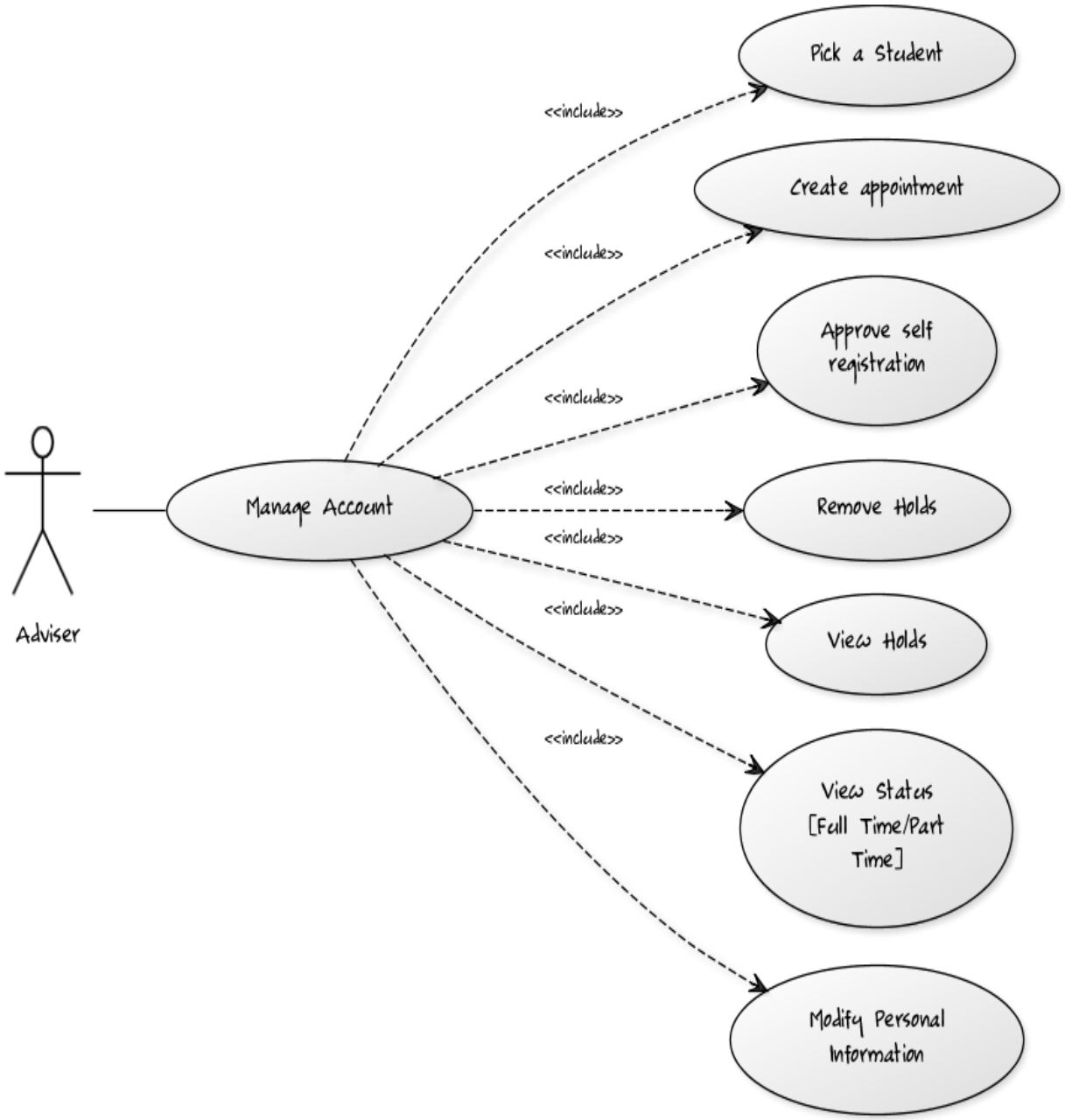


Brief Description:

The *Manage Class Schedule* use case will allow an Adviser to view and edit several different things on advising student account.

Step-by-Step Description:

1. Adviser can view student's Class Detail
2. Adviser can Print Out Schedule of Classes



Brief Description:

The ***Manage Account*** use case will allow a Adviser to view and edit several different things on Student's account.

Step-by-Step Description:

1. Adviser can remove holds on Student account.
2. Adviser can view holds on Student account.
3. Adviser can view Student enrollment status using the *View Status [Full Time/Part Time]* use case.
4. Adviser can edit Student account information using the *Modify Personal Information* use case.
5. Adviser can create appointment with Student
6. Adviser can pick a student to be his/her adviser
7.Adviser can approve self registration for new Students

Brief Description:

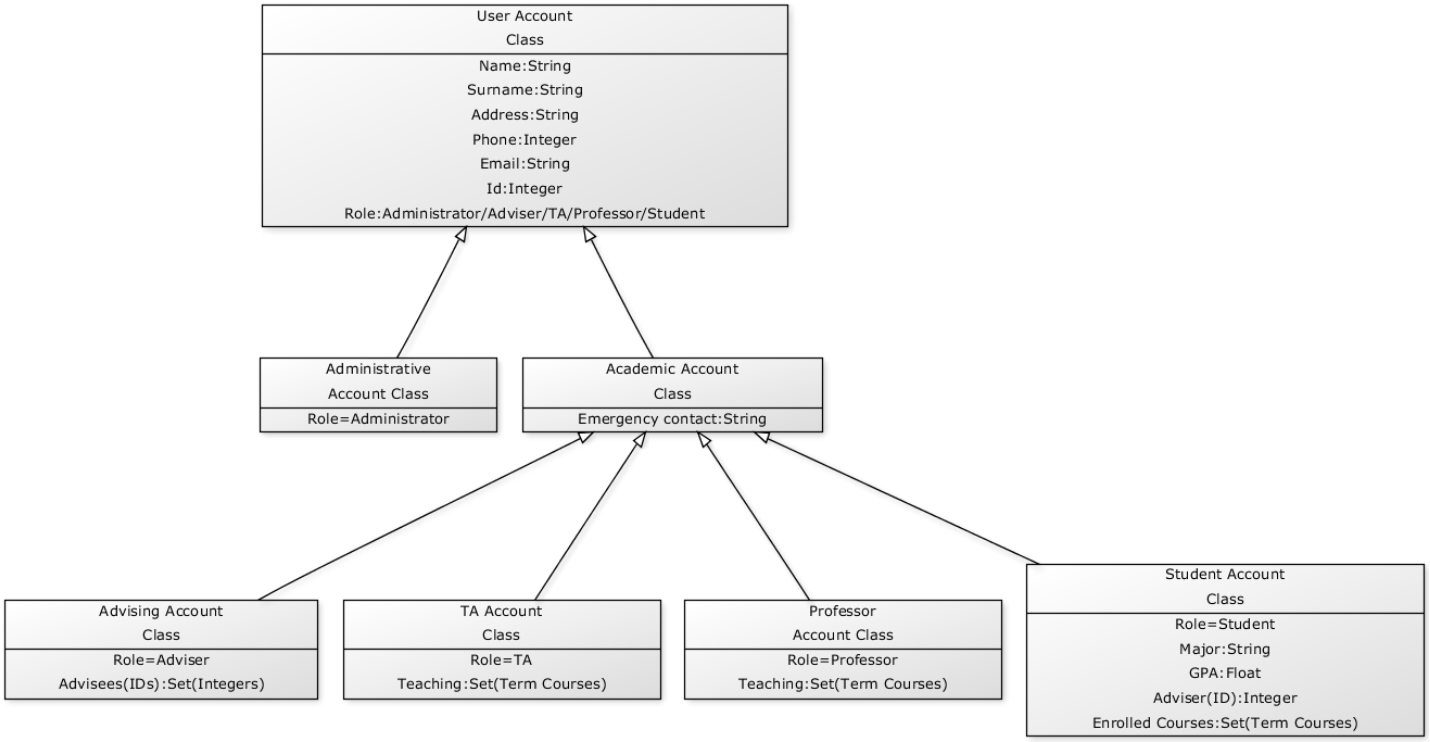
The ***Manage Graduation*** use case allows a adviser to to view student graduation audit report.

Step-by-Step Description:

1. Adviser can view Student's graduation audit report using the *View Graduation Audit* use case.

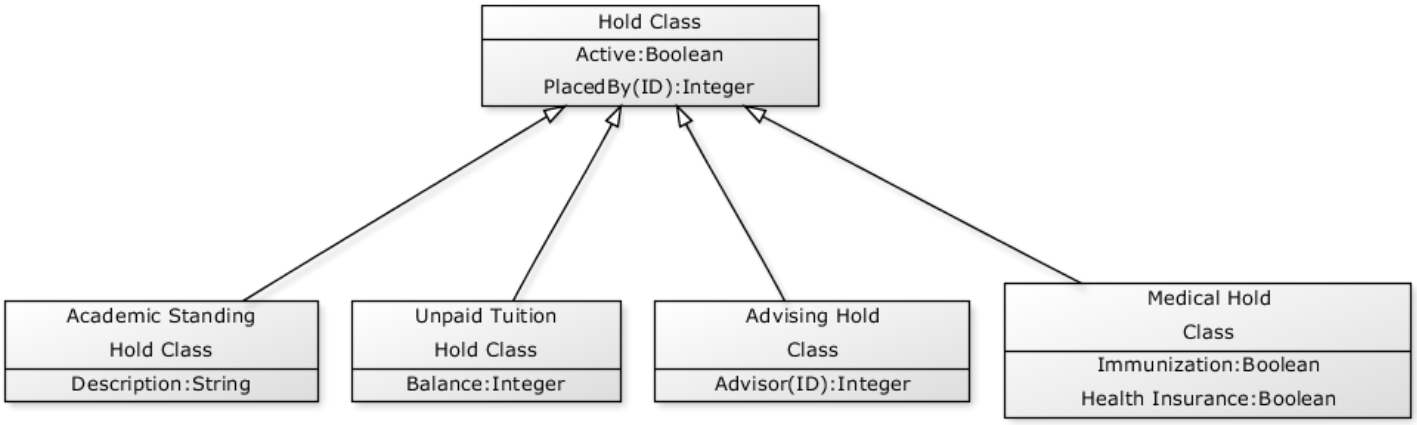
4. Extraction of classes given as detailed class diagrams

4.1 Entity Classes



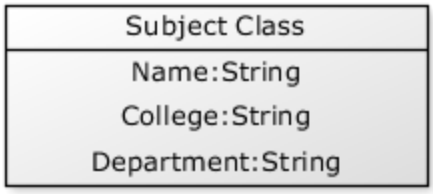
User Account:

User Account is an abstract class. Each class that implements User Account will contain: Name, Surname, Address, Phone, Email, ID, and the Role. The subclasses are: Administrative Account Class and Academic Account Class. The administrative account is a concrete class assigned to role: administrator. Academic Account is an abstract class and includes an emergency contact attribute. Its subclasses are: Advising Account, TA Account, Professor, and Student Account, and the classes are assigned to the adviser, ta, professor, and student roles, respectively. The Advising Account Class contains a set of advisees by their IDs. The TA and Professor Account Classes both contain a set of courses being taught by the instance class. The Student Account Class contains the Major, GPA, Adviser, and set of enrolled courses for current term.



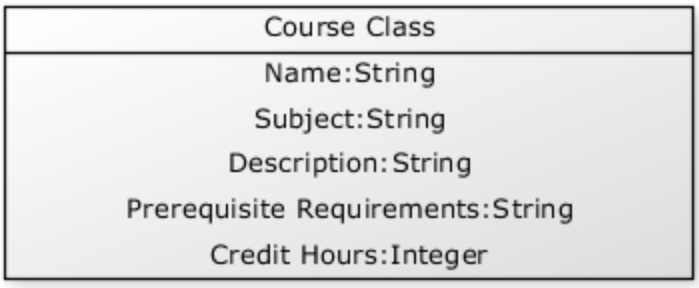
Hold Class:

The Hold Class is an abstract class that includes a boolean value representing whether the particular hold is active and an ID Integer value of the user who placed the hold. There are four classes that implement the Hold Class: Academic Standing Hold, Unpaid Tuition Hold, Advising Hold, and Medical Hold. The Academic Standing Hold Class contains a description of the nature of the hold. Unpaid Tuition Hold Class contains the unpaid balance ammount. The Advising Hold Class contains the assigned Adviser's ID. Medical Hold Class contains two boolean values that signify whether the hold stems from missing Immunization or Health Insurance.



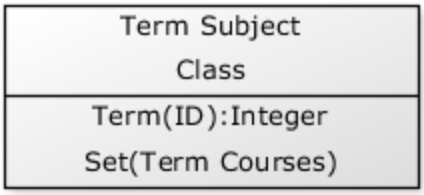
Subject Class:

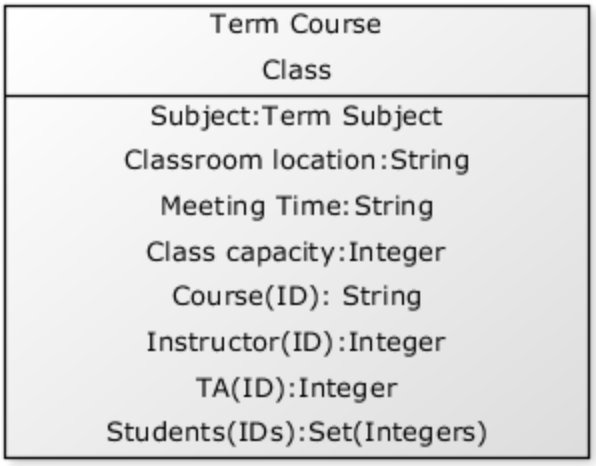
The Subject Class contains the subject's Name, College, and Department. A subject class represents subject details that go unchanged between terms.

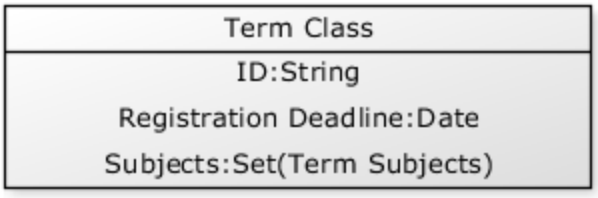


Course Class:

The Course Class contains the course Name, Subject, Description, Prerequisites, and number of Credit Hours. A course class represents course details that go unchanged between terms.







Term Subject Class:

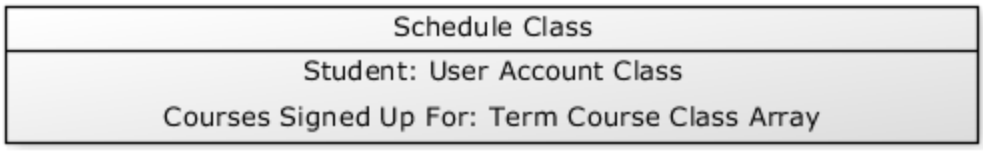
A Term Subject Class includes all of the attributes of a Subject Class and also contains the Term ID to which it belongs and a set of Term Courses being offered under the Term Subject instance. A term subject represents all courses being offered in a term for the particular subject.

Term Course Class:

A Term Course Class includes all of the attributes of a Course Class and also contains the term Subject it belongs to, the Classroom location, Meeting time, Class capacity, Course ID, ID of Instructor and TA, and a set of Students' IDs enrolled in the course. A term course represents the course details for the particular term.

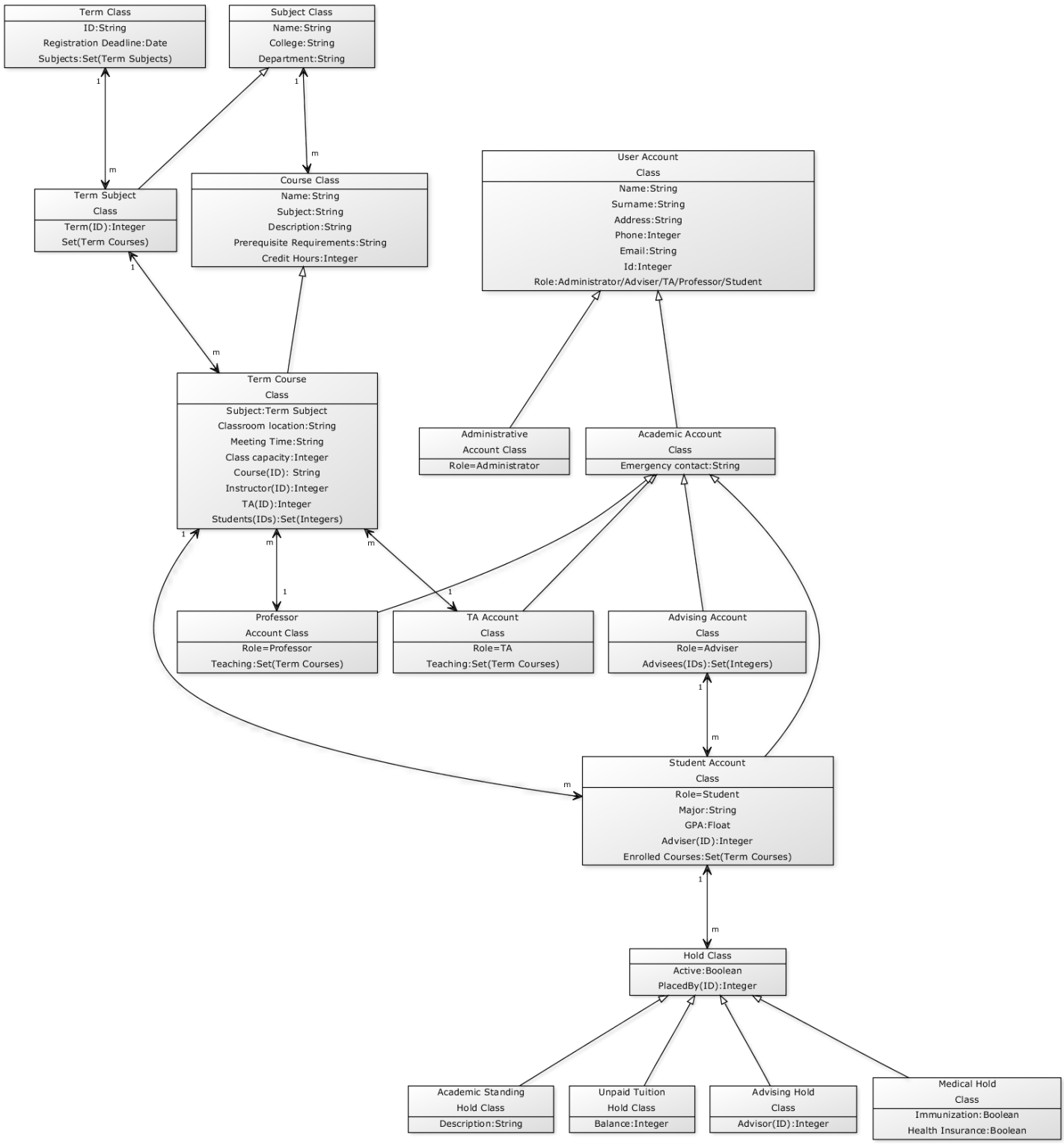
Term Class:

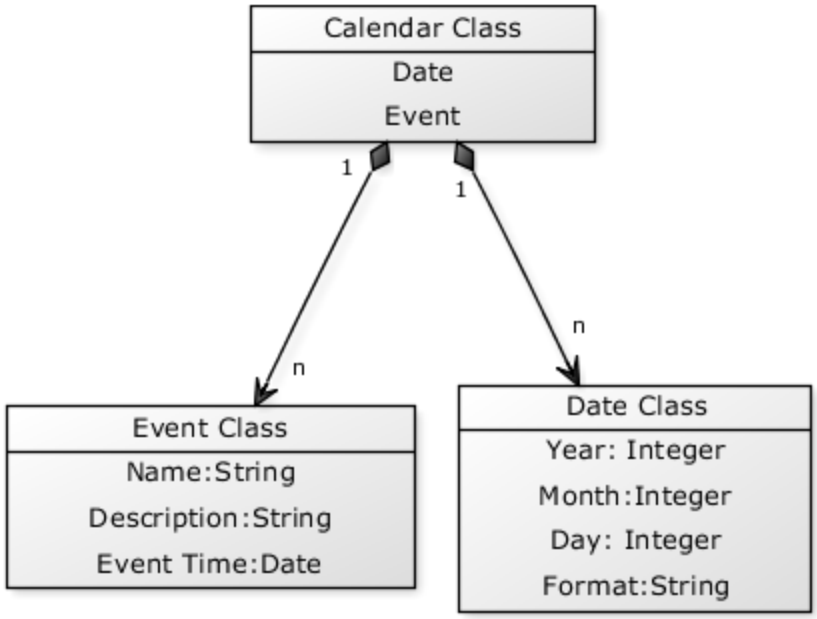
A Term Class contains the term ID, Registration Deadline, and a set of Subjects in the particular term. A term (or semester) represents all subjects being offered.



Schedule Class:

This class allows a particular schedule to be associated to a Student long term in the software life cycle.


Term Class
ID:String
Registration Deadline:Date
Subjects:Set(Term Subjects)
Subject Class
Name:String
College:String
Department:String
Term Subject Class
Term(ID):Integer
Set(Term Courses)
Course Class
Name:String
Subject:String
Description:String
Prerequisite Requirements:String
Credit Hours:Integer
User Account Class
Name:String
Surname:String
Address:String
Phone:Integer
Email:String
Id:Integer
Role:Administrator/Adviser/TA/Professor/Student
Term Course Class
Subject:Term Subject
Classroom location:String
Meeting Time:String
Class capacity:Integer
Course(ID): String
Instructor(ID):Integer
TA(ID):Integer
Students(IDs):Set(Integers)
Administrative Account Class
Role=Administrator
Academic Account Class
Emergency contact:String
Professor Account Class
Role=Professor
Teaching:Set(Term Courses)
TA Account Class
Role=TA
Teaching:Set(Term Courses)
Advising Account Class
Role=Adviser
Advisees(IDs):Set(Integers)
Student Account Class
Role=Student
Major:String
GPA:Float
Adviser(ID):Integer
Enrolled Courses:Set(Term Courses)
Hold Class
Active:Boolean
PlacedBy(ID):Integer
Academic Standing Hold Class
Description:String
Unpaid Tuition Hold Class
Balance:Integer
Advising Hold Class
Advisor(ID):Integer
Medical Hold Class
Immunization:Boolean
Health Insurance:Boolean

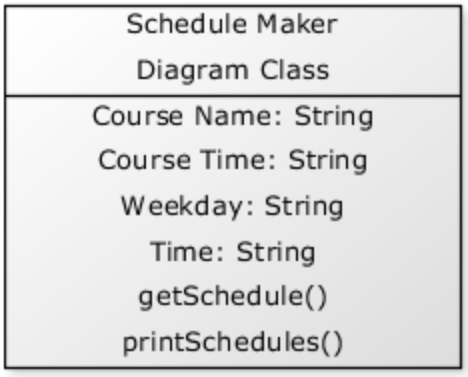


Calendar Class

The Calendar Class maintains the list of all the events. If someone wants to assign a particular event, He passes the event to the calendar class. The class will first confirm whether the event can be scheduled on the specific date. Only after verifying that no other event is taking place on that day, the class will assign that event on the asked date. Otherwise, it will send proper message to the event class.

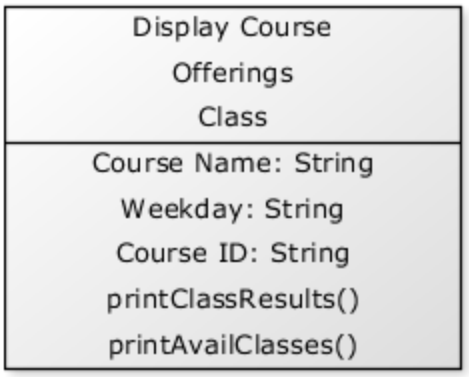
[Message Class]
[Message|Sender:String;Receiver:String;Subject:String;Body:String;sendDate:Date;readDate:Date]

4.2 Boundary Classes



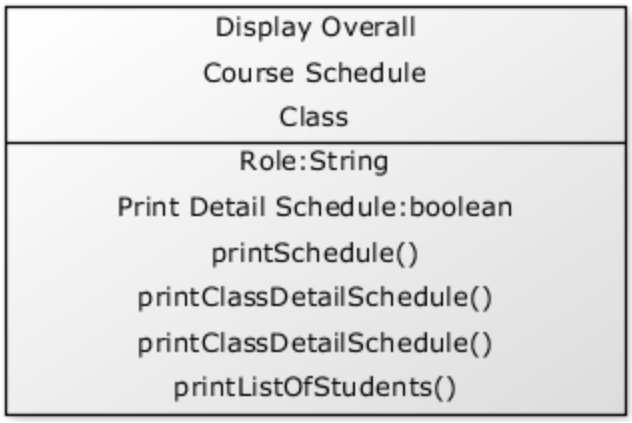
Schedule Maker Diagram Class:

Generates a GUI for the students to interact with the Schedule Maker functionality. It will display schedules and provide several input menus.



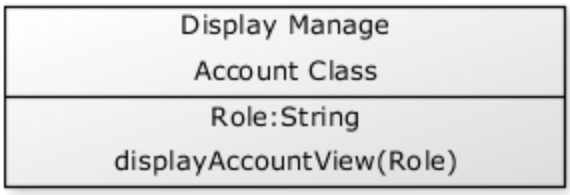
Display Course Offerings:

Provides an easy to read GUI for Students searching for classes to enroll for in the semester.



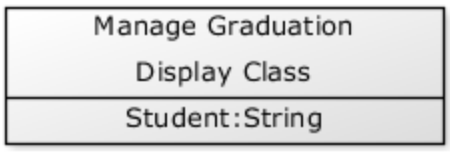
Display Overall Course Schedule Class:

Provides a GUI for the Student, Professor, TA, and Adviser to view, print, and provide inputs for schedules generated.



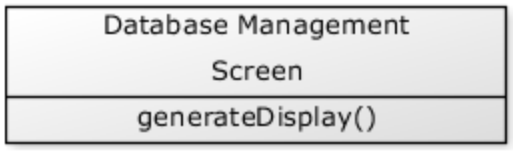
Display Manage Account Class:

Generates the GUI for the Student, Professor, TA, Adviser, and Admin to manage their own accounts or another user's account depending on their permissions.



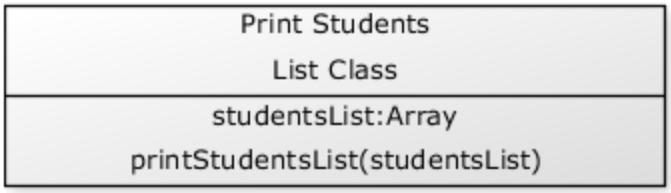
Manage Graduation Display Class:

Creates the GUI for Students to view their graduation audit information and apply for graduation.



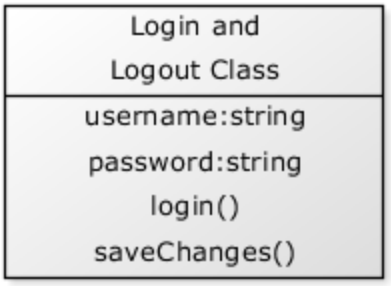
Database Management Screen:

Gives the GUI interface for the Admin to perform several administrative actions with the database.



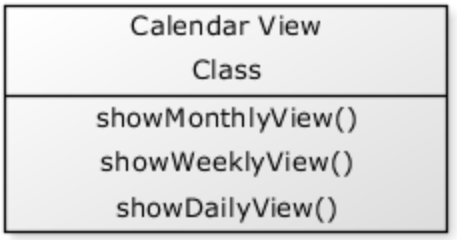
Print Students List Screen:

The given function takes a list of students and prints out a list if requested.



Login Screen:

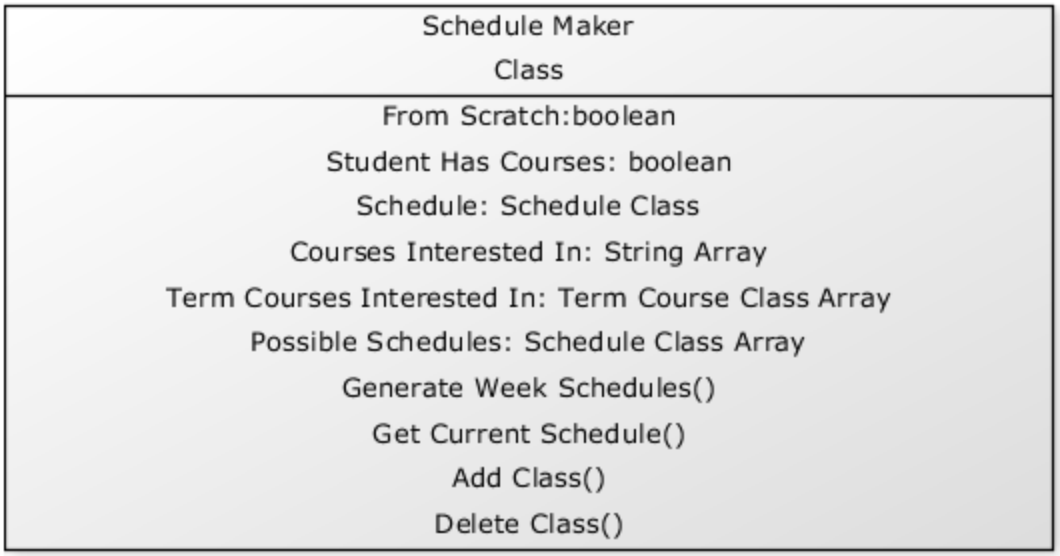
The Login Screen will be generic for all the users. The users will give their username / password and after authentication that will be able to enter the application. Once the users are done they can select the logout option.



Calendar View Class:

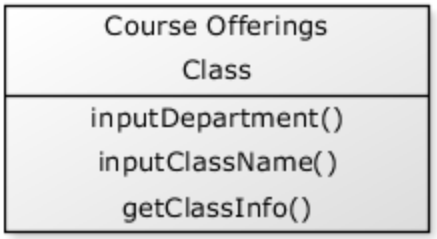
Using this class, the user will be able to view the calendar in either the monthly, weekly or the daily mode.

4.3 Control Classes



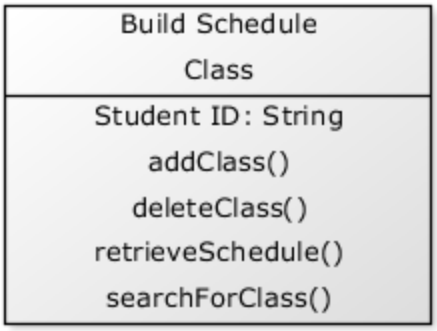
Schedule Maker Class:

This class will aid Students in generating a schedule from scratch or one with classes that can fill open slots in their schedule. A Student can provide up to 5 classes they are interested in taking and the class will generate a list of schedules that are possible. A greedy algorithm for class selection will be used to generate the possible schedules.



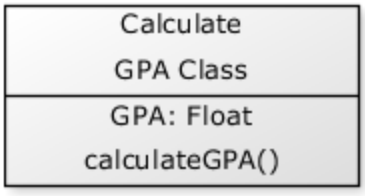
Course Offerings Class:

Facilitates in the searching of classes offered during the current term.



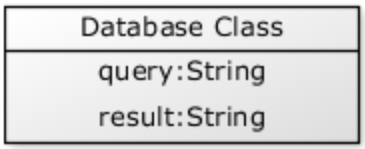
Build Schedule Class:

This class will help create and store a Student's schedule for the semester.



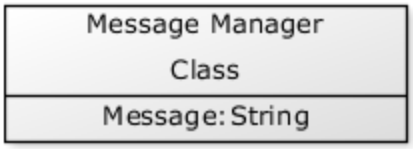
Calculate GPA Class:

Calculates GPA for a student based on a 4.0 scale. Letter grades A, B, C, D, F are assigned values 4.0, 3.0, 2.0, 1.0, 0.0 respectively. Also, each course registered for by the Student is assigned a certain number of credit hours. Each course grade is multiplied by the number of credit hours and summed. Divide this sum by the sum of the credit hours. The resulting is the GPA for the Student.



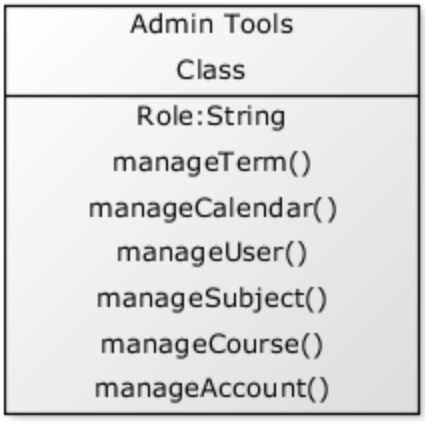
Database Class:

This is the main class for interacting with the database. Queries, edits, and other operations are made through this class.



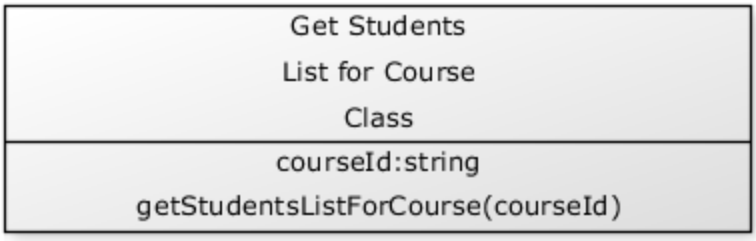
Message Manager Class:

This class allows communication (sending of messages) between the several users. The main method for implementing this, will be a simple write/read to and from the database.



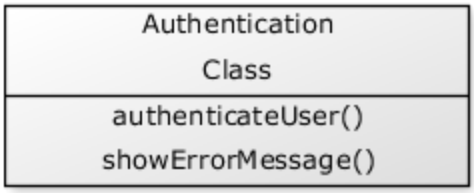
Admin Tools:

This class will allow different users to manage several different settings, accounts, and other administrative procedures depending on their permissions.

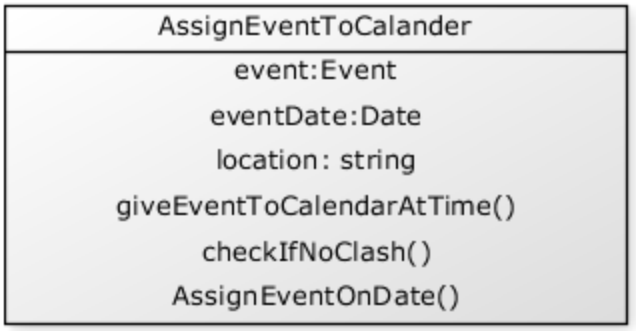


Get a List of Students:

This class will allow different users to give a course ID and they will be given the list of students taking that course.

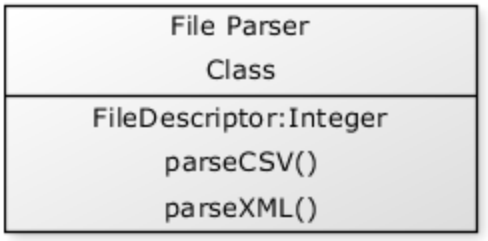


Authentication Class
This class authenticates the user by taking the username and password of the user. After verification from the database, the class will let the user in the system. Otherwise, error message will be displayed to the user.



AssignEventToCalendar

Using this class, the user who wishes to assign an event, will be able to assign an event on a particular day at a particular time. The class will check if the slot is free for the event to take place and will assign or give proper message accordingly.



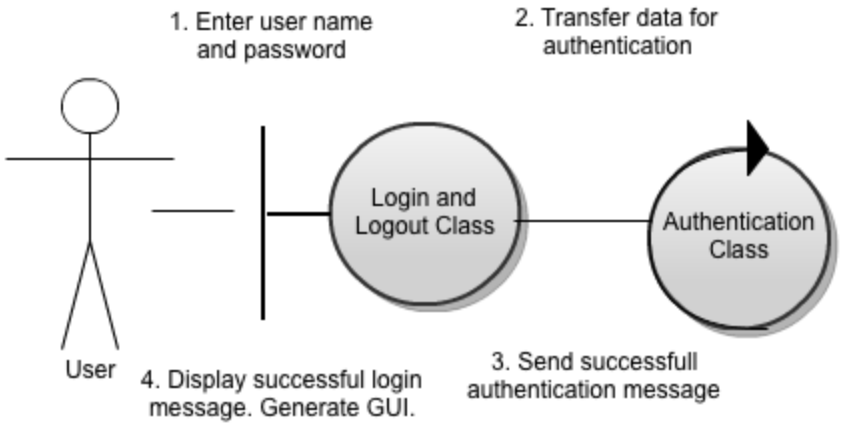
File Parser Class

The File Parser Class contains the utility helper functions for parsing a csv or xml data file.

5. Use Case Realizations

Users:

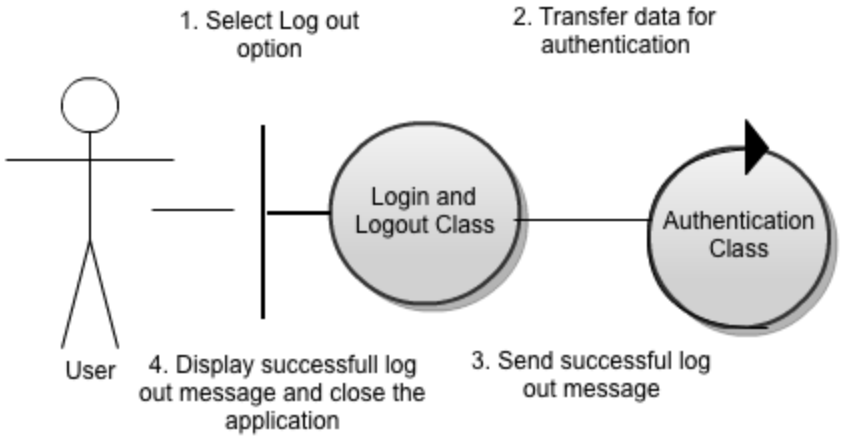
User will be defined as any Student, Professor, Teacher Assistant, Adviser, or Administrator. This Use Case Realization is common to all users, and hence, represented just once.



Senario 1:

User wants to use the application.

1. The user will open up the application.
2. A log in screen will be generated by the System.
3. The User will enter their user name and password.
4. The System will authenticate the User and allow them access to the application.
5. GUI will be generated by the System for the User.

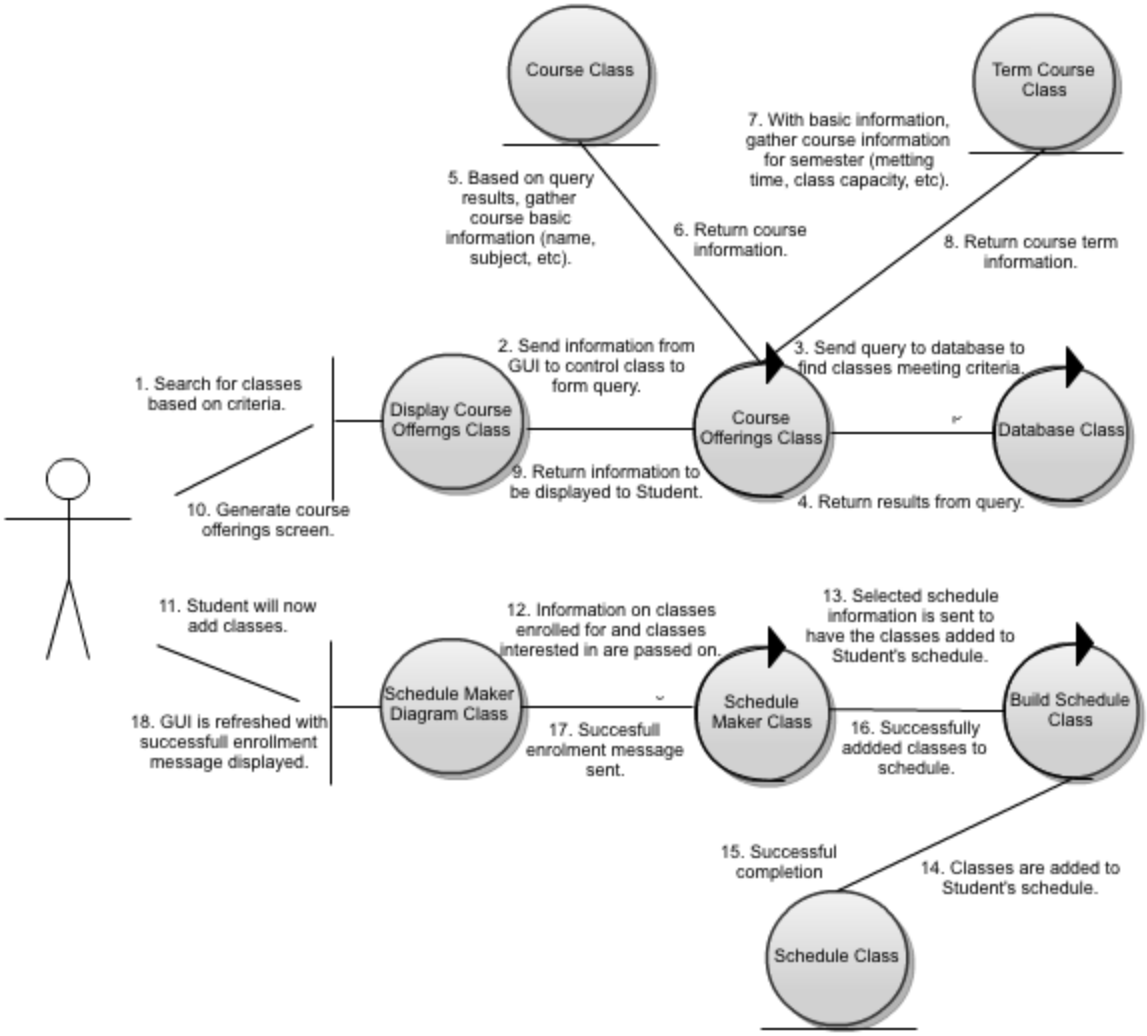


Scenario 2:

User wants to exit the application.

1. Once the User is done using the system they will select the log out option.
2. The System will log the user out and close the application.

Student:

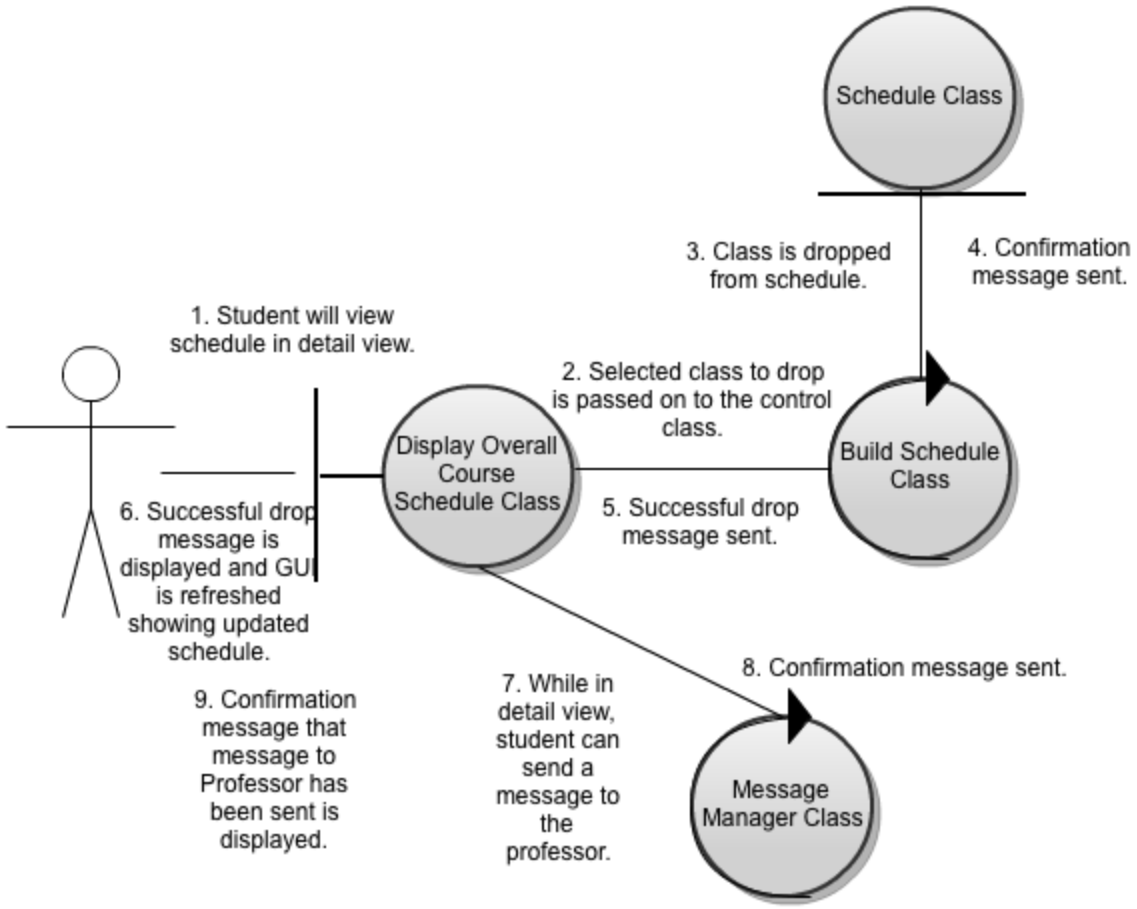


Scenario 1:
Student wants to look up classes and register for the term.

1. The Student will navigate through the menu to get to a list of courses offered for the semester.
2. The Student will look up the Classes offered this semester based on:
    - Term
    - Subjects name

- Course name

3. The courses offered this term will be displayed by name
4. Continuing to use the menu, the student select a course he/she is interested in taking.
5. By default the system will show classes with seats still available
6. The student can use the GUI to have either only closed or all (open and closed) classes displayed.
7. If a class is closed the student will have the option of getting on a waiting list.
8. The system will show a quick view displaying the following information:
    - Course name
    - Course ID
    - Seats Left
    - Meeting Time
9. The Student will add a class to his/her schedule by navigating to the Schedule Maker screen
10. Using the GUI the student will enter the classes he/she is interested in taking. The total credit hours must be enough to meet full time status (12hrs) or part time status (<12hrs)
11. If the Student is already enrolled in classes, the Schedule maker will show a set of possible schedules using the classes given to fill the empty time slots in the Students schedule.
12. If the Student is not enrolled for any classes, the Schedule maker will show a set of possible schedules.
13. The Student will Select a schedule and be enrolled for the classes.



Scenario 2:

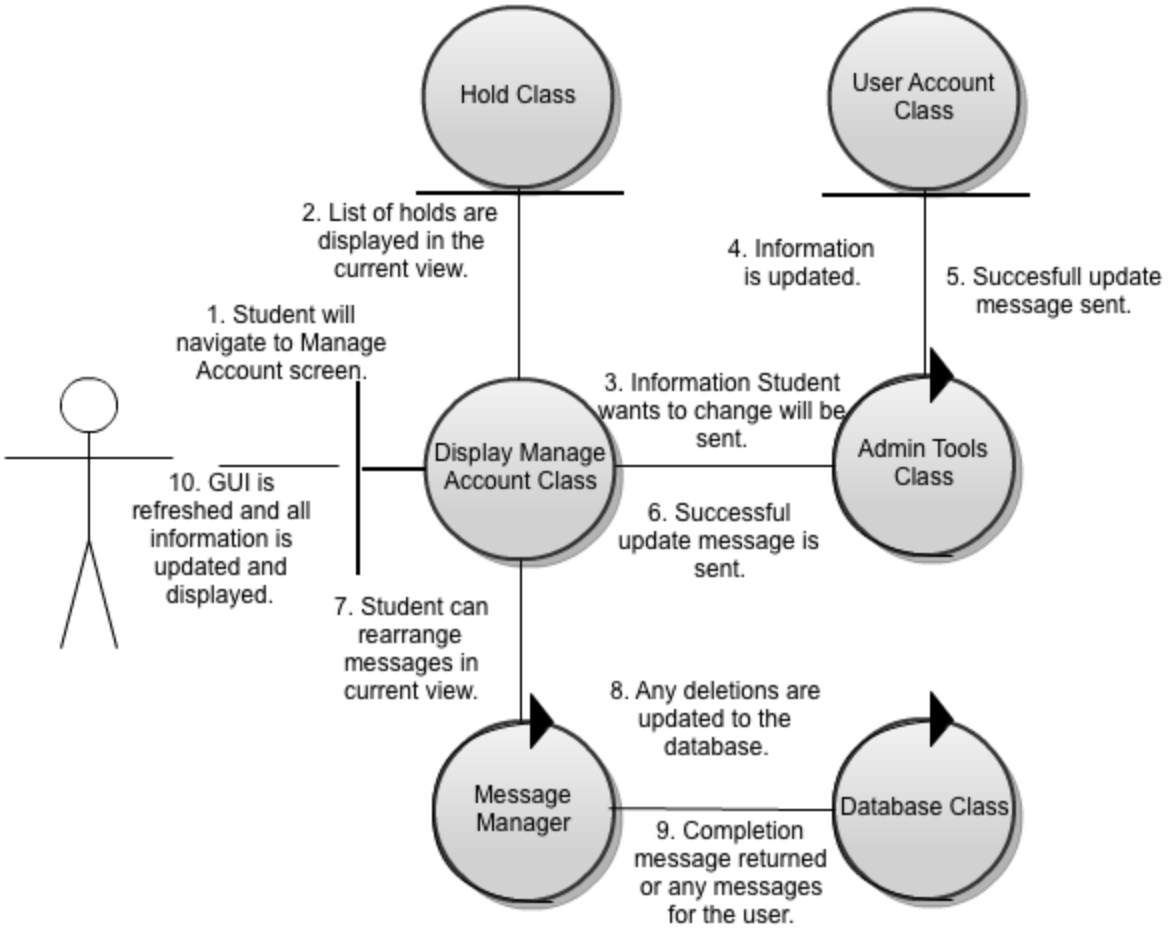
Student wants to view his current schedule, drop a class, and print his revised Schedule.

1. Student will navigate to the GUI for viewing his overall class schedule.
2. The System will display a graphical view of his current schedule by default. The display will be in a calendar week format.
3. Class details shown will be as follows [Term Course Class]
    - Name
    - Meeting time
    - Classroom location

4. The Student can choose to change the view to a detailed class view. This view will show further details than the default view.
5. Class details will be as follows: [Term Course Class]
    - Name
    - Meeting time

- Classroom location
- Course ID
- Instructor
- TA

6. In detail view the Student can send the Professor a message [Message Manager]
7. Student will drop a class
8. In detail view, the student can select a class to drop.
9. The system will remove the class from the Student's schedule and refresh the GUI.
10. Student can select to print their revised schedule in either weekly or detail view.

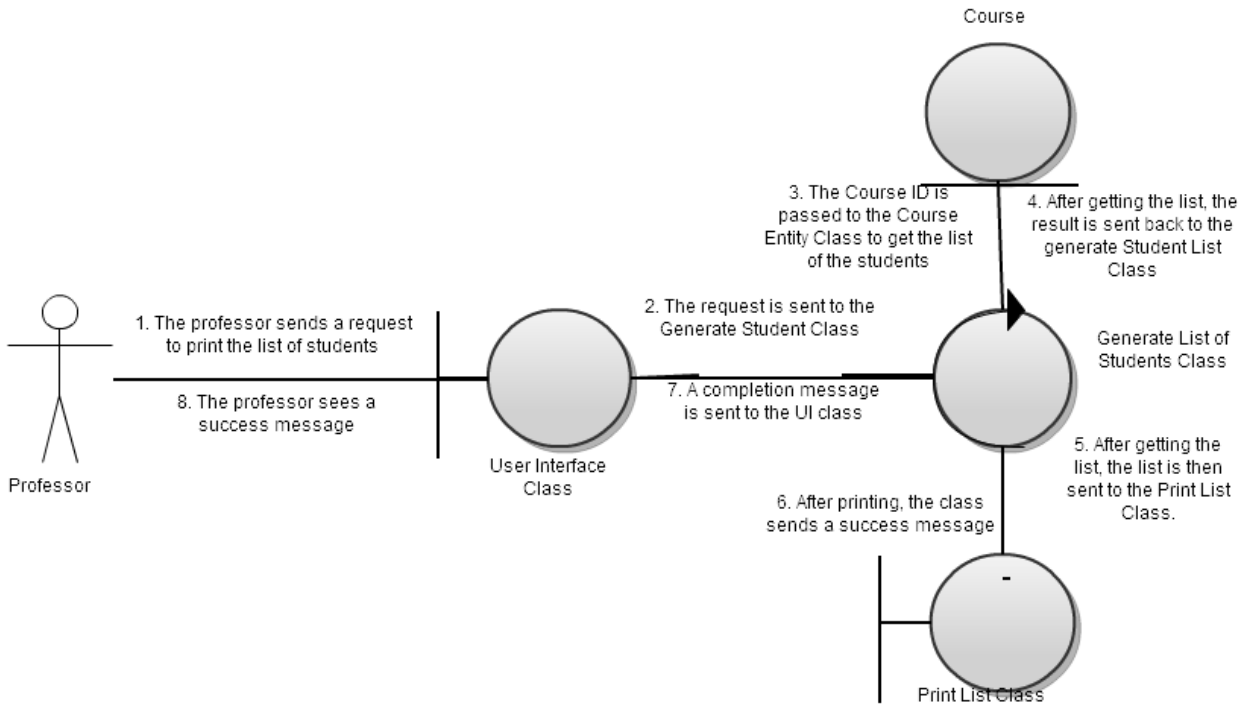


Scenario 3:

The Student wants change their address, make sure there are no holds on their account, review any messages, make sure he/she is enrolled as full time status, review their graduation audit, and apply for graduation.

1. The Student will navigate to the Manage Account Screen.
2. There, the system will display the Student's basic account information:
    - Name
    - Student ID
    - Address
    - Phone number
    - Full Time/ Part Time status
    - Holds on his/her account
    - Messages
3. While in this view, the student can arrange the Messages received in several different formats:
    - Sender (alphabetically)
    - Importance
    - Date
4. The Student can opt to change his personal information via the GUI
5. The Student will only be able to change
   Address
   Phone Number
6. The Student can then navigate to the Manage Graduation GUI
7. The Student can have a graduation audit report generated by the system using the GUI.
8. The system will display the graduation audit to the Student
9. Once the graduation audit shows that the student has completed all requirements, the student can use the GUI to apply for graduation.

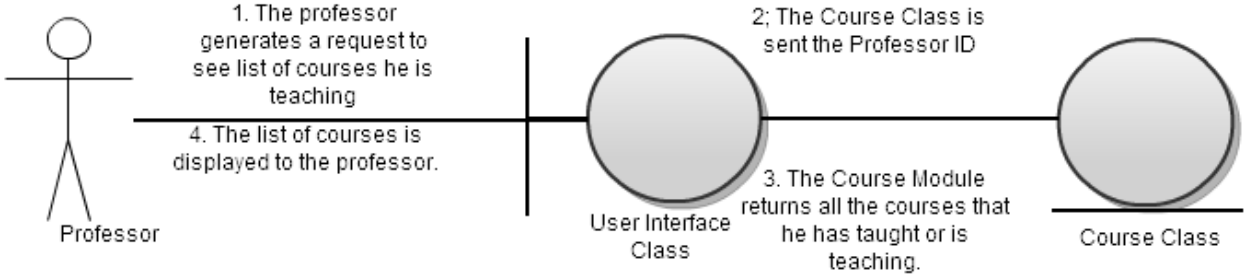
Professor:



Scenario 1:

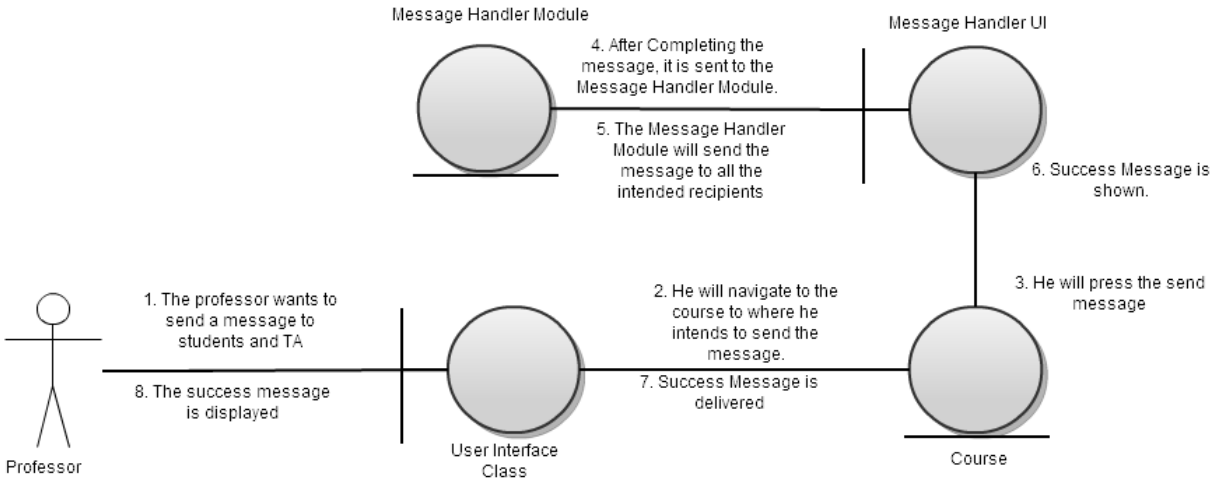
The Professor wants to print a list of students taking his course.

1. The Professor will navigate to the current courses he is teaching screen.
2. The Professor will select the course of which he wants to get the list.
3. The request will go to the course module which will then pass the list of students that are taking that particular course.
4. After the list of students is displayed, he will click the print button and the list will print.



Scenario 2:
The Professor wants to check which classes he has been assigned to and wants to check whether or not he has previously taught the courses.

1. The Professor will navigate to Manage Account screen.
2. The system will display a list of courses that the Professor has been assigned to for the current semester.
3. The Professor can use the GUI to generate the list of courses that he has taught previously.



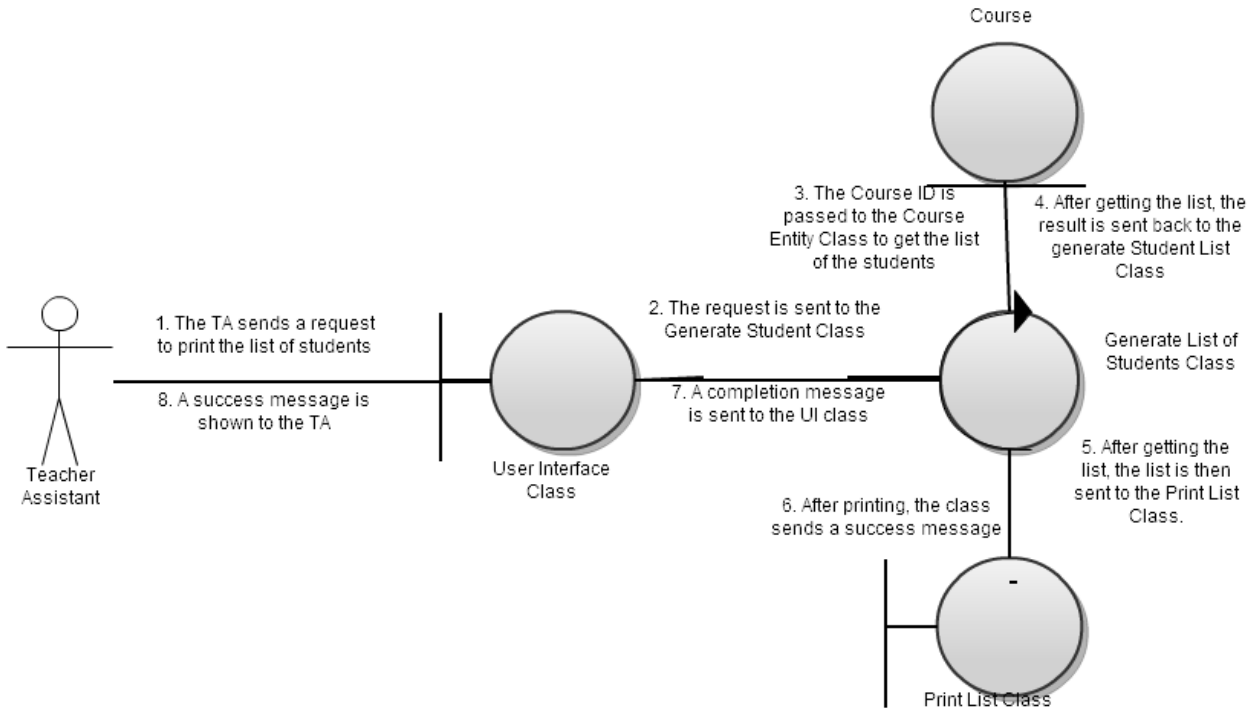
-

Scenario 3:
The Professor wants to send a messages to both the Students and TA.

1. Professor will navigate to the list of courses he is teaching this term.
2. He will select the course to whose students he wants to send a message to using the GUI.
3. The Professor will then write down the message and click send.

4. The System [Message Handler] will deliver the message to all the recipients of the message.
5. The System will refresh the GUI and now the Professor will select the TA he wants to send a message to.
6. The Professor will then write down the message and click send.
7. The System [Message Handler] will deliver the message to the TA.

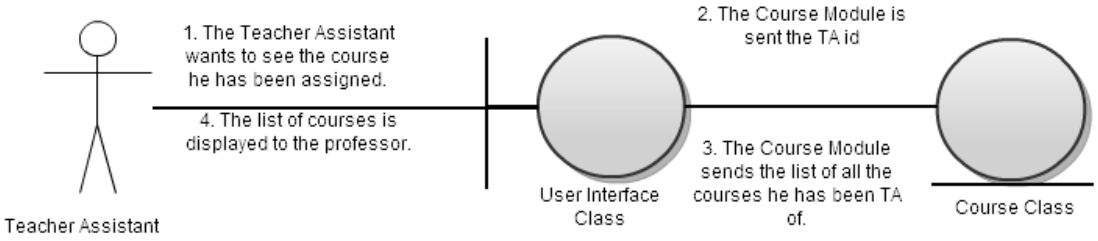
Teacher Assistant:



Scenario 1:

The TA wants to print a list of students for a course he is assisting this semester.

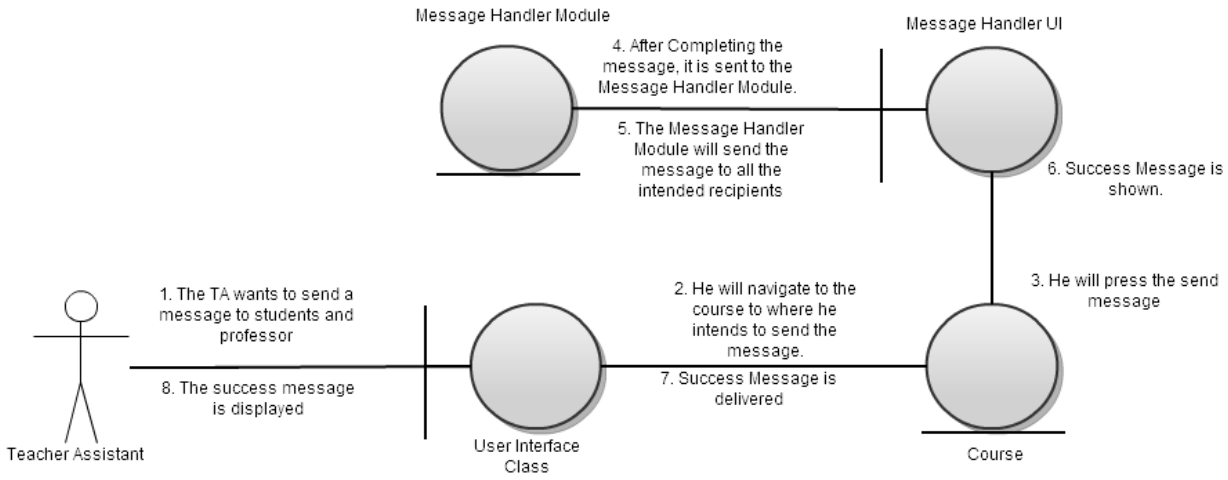
1. The TA will navigate to the current courses he is teaching screen.
2. The TA will select the course of which he wants to get the list.
3. After the list of students is displayed, he will click the print button and the list will print.



Scenario 2:

The Teacher Assistant wants to check which courses he has been assigned to and if he has assisted in this course before.

1. The TA will navigate to Manage Account screen.
2. By default the System will generate the list of courses that he has been assigned to.
3. The TA can use the GUI to generate the list of courses that he has taught previously.



-

Scenario 3:

Teacher Assistant wants to send a message to Students and Professor.

1. TA will navigate to the list of courses he is assisting this term.
2. He will select the course to whose students he wants to send a message to using the GUI.
3. The TA will then write down the message and click send.
4. The System [Message Handler] will deliver the message to all the recipients of the message.
5. The System will refresh the GUI and now the TA will select the Professor he wants to send a message to.
6. The TA will then write down the message and click send.
7. The System [Message Handler] will deliver the message to the Professor.

Adviser:

Scenario 1:

Adviser wishes to manage students account.

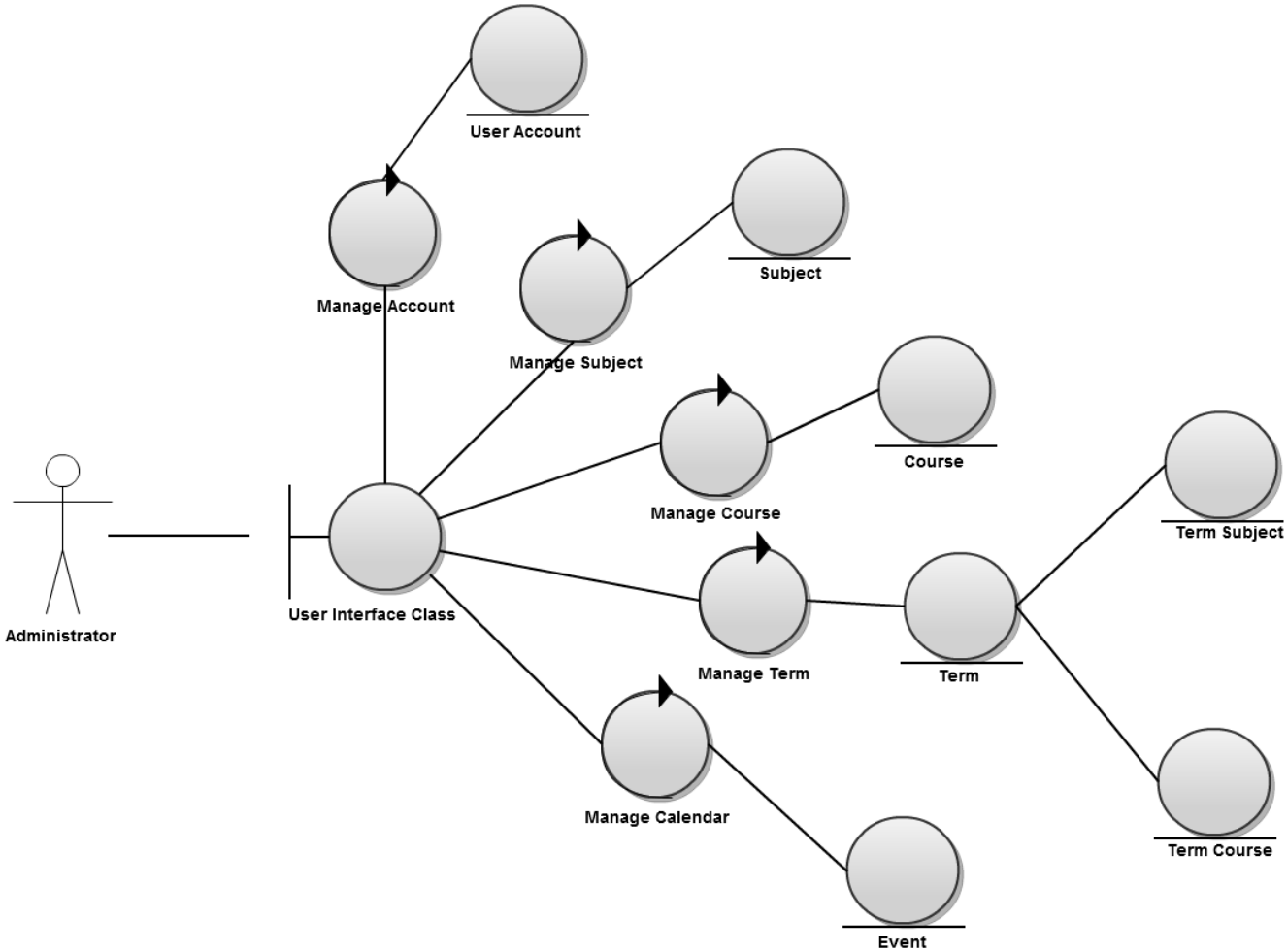
1. The Advisers navigates to list of all students
2. System displays list of all students.
3. Adviser selects a student to manage
4. The system gives Adviser several options to choose from: pick a student, create appointment, approve self-registration, remove hold, view holds, view status, modify personal information.
    - 4.1 Adviser selects “pick a student”
        - 4.1.1 System will register Adviser choice and send a message to student about his/her new adviser.
    - 4.2 Adviser selects “create appointment”
        - 4.2.1 Adviser enters desired date and time for the appointment.
        - 4.2.2 System stores appointment and sends confirmation to the student.
    - 4.3 Adviser selects “approve self-registration”
        - 4.3.1 System checks if student is newly registered or already approved
        - 4.3.2 System sends message to the student about Adviser’s approval
    - 4.4 Adviser selects “remove hold”
        - 4.4.1 System displays possible holds on students account
        - 4.4.2 System prompts Adviser to remove existing hold on student’s account.
    - 4.5 Adviser selects “view holds”
        - 4.5.1 System generates a list of holds on student account
    - 4.6 Adviser selects “view status”
        - 4.6.1 System displays student status: full time, part time
    - 4.7 Adviser selects “modify personal information“
        - 4.7.1 System retrieves the user data of the selected student and displays it to the Adviser in a modifiable form.
        - 4.7.2 Adviser makes changes to data fields in the form and submits.
        - 4.7.3 System updates the user account.
        - 4.7.4 System sends message to the student that his account was modified by a Adviser.
5. System updates database to adjust new changes.

<u>Scenario 2:</u>
An Adviser wishes to approve new student registration

1. Adviser will log into the system and be authorized with adviser privileges.

2. Adviser will click on the list of new registered students
3. System will return a list of new registered students
4. Adviser will select a student for approval
5. System will return student info for Adviser to edit/approve
6. Adviser will have an option for approve/disapprove student.
7. After Adviser approves a new student system will record new student and change its privileges to a active student (able to register for classes)
8. System sends a message to the new student about Advisers approval.
9. Adviser is asked to put new student under his advising list.

Administrator:

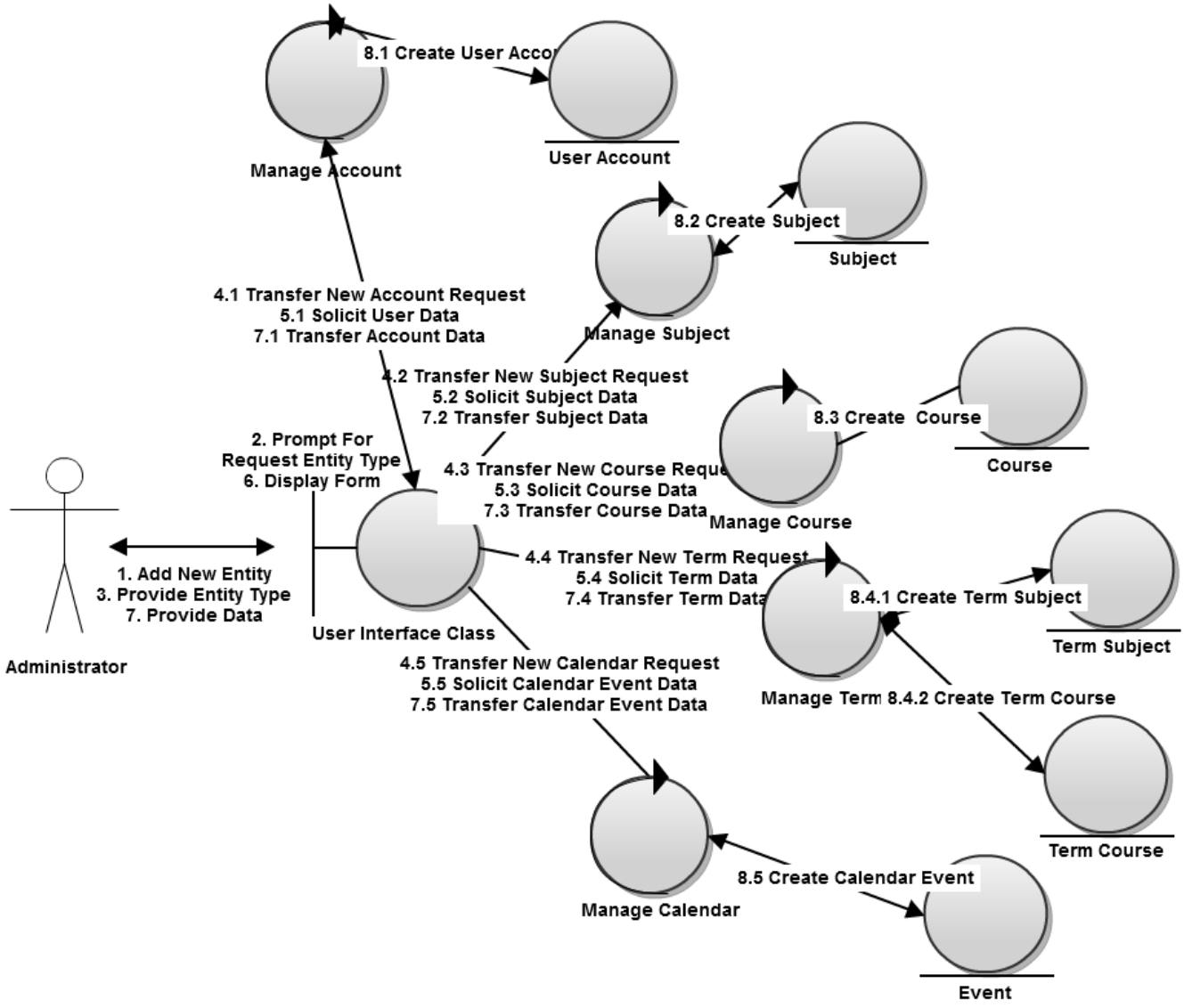


Administrator Class Diagram

Scenario 1:

An Administrator wishes to add a new entity into the system.

1. The administrator navigates to add new entity
2. The system prompts the user to select one of the following options: add new user account, subject, course, term, or calendar event to the system.
3. The administrator selects an option.
    - 3.1 “add new user account”.
    - 3.2 “add new subject”.
    - 3.3 “add new course”.
    - 3.4 “add new term”.
    - 3.5 “add new calendar event”.
4. The system displays a form for adding the appropriate entity and requests pertinent data from the administrator.
5. The administrator submits data to add the new entity.
6. The system creates the new entity.



Communication Diagram of realization of Scenario 1

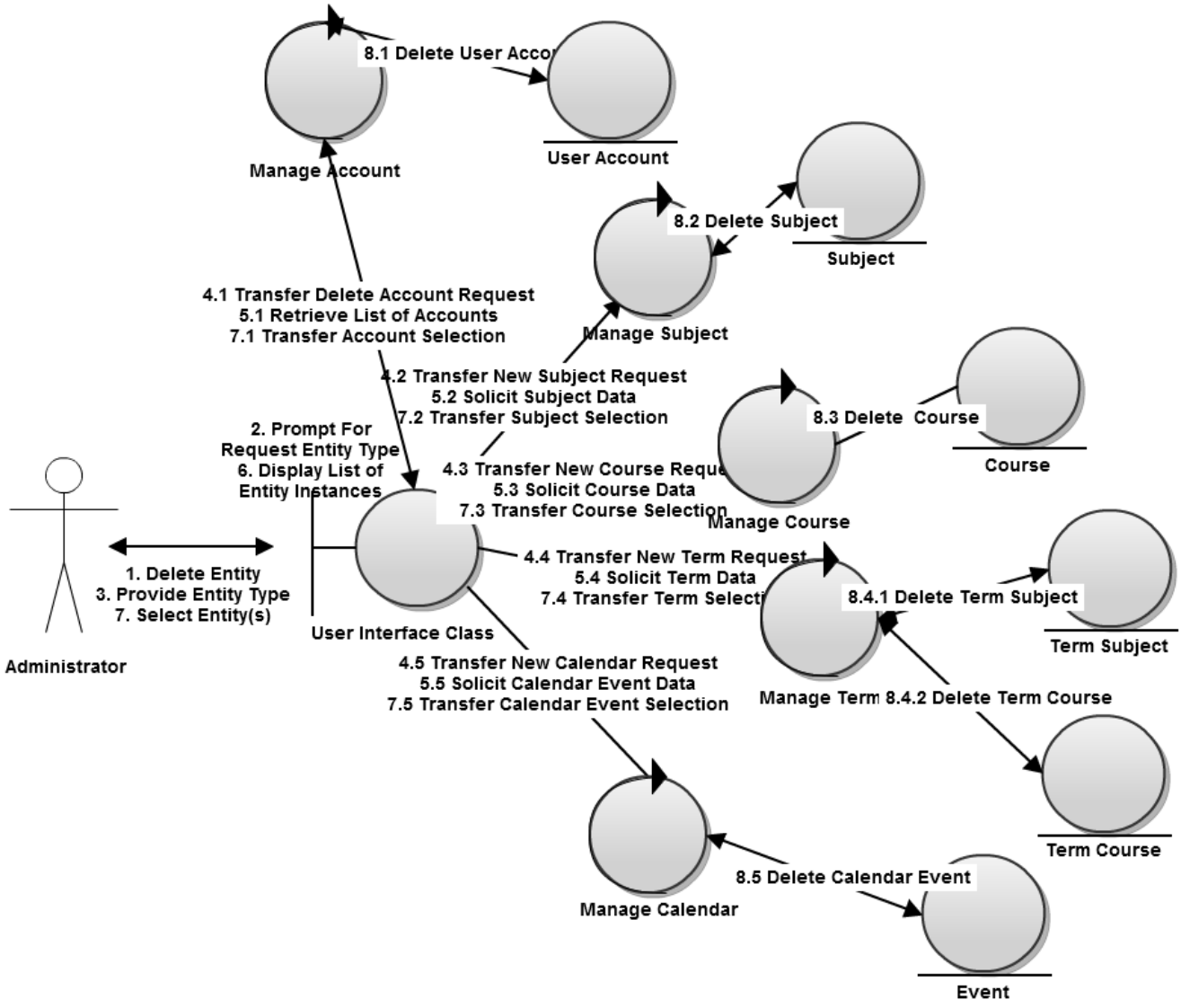
Scenario 2:

An administrator wishes to delete an entity from the system.

1. The administrator navigates to delete existing entity.
2. The system prompts the user to select one of the following options: remove existing user, subject, course, term, or calendar event from the system.
3. The administrator selects an option.

3.1 “delete existing user account”.
3.2 “delete existing subject”.
3.3 “delete existing course”.
3.4 “delete existing term”.
3.5 “delete existing calendar event”.

4. The system retrieves a list of all appropriate entities and displays it back to the admin.
5. The admin selects one or more entities to remove and submits the request.
6. The system deletes the entity(s)

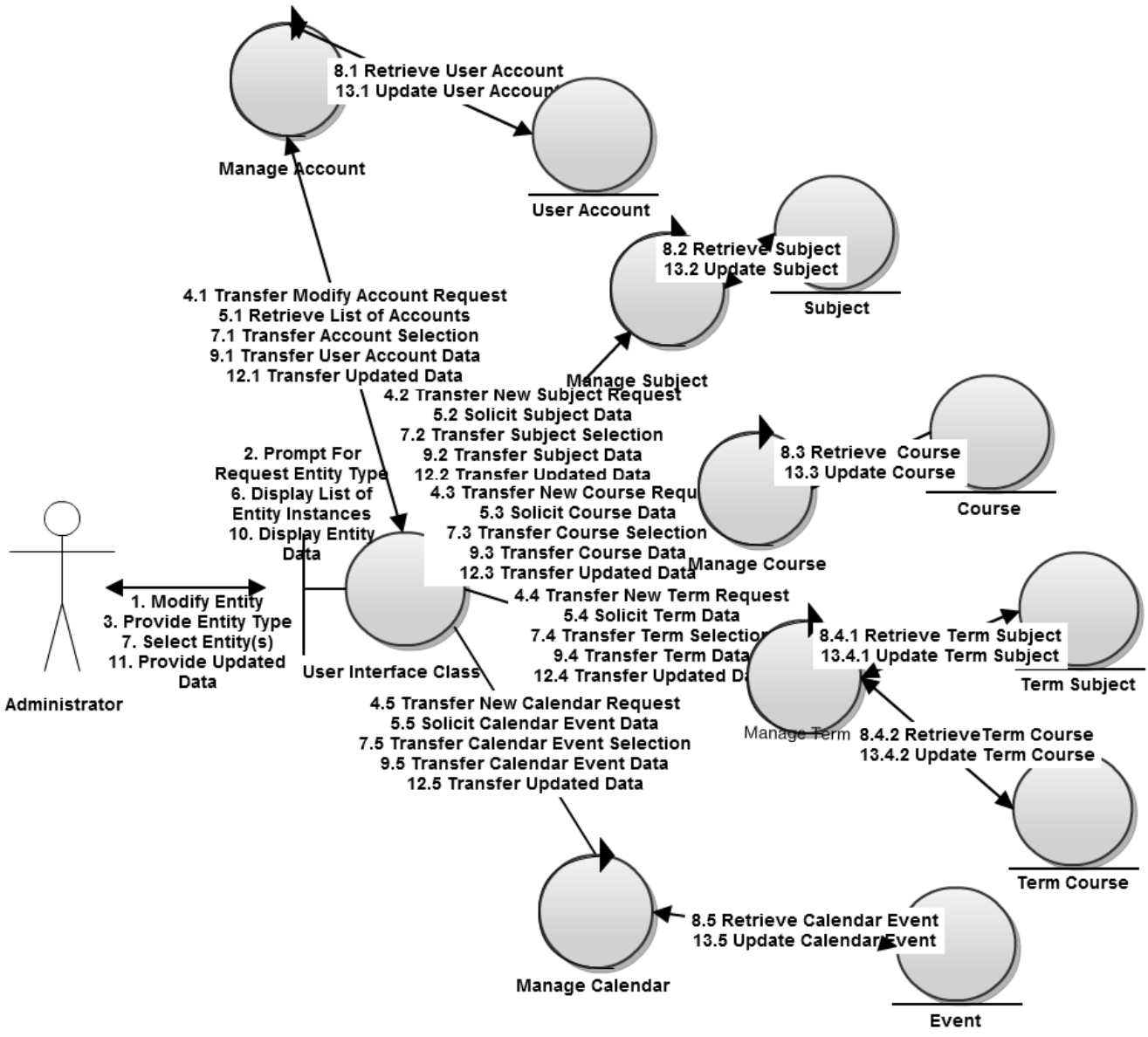


Communication Diagram of realization of Scenario 3

Scenario 3:

An administrator wishes to modify an existing entity.

1. The administrator navigates to modify existing entity.
2. The system prompts the user to select one of the following options: modify existing user, subject, course, term, or calendar event from the system.
3. The administrator selects an option.
    - 3.1 “modify existing user account”.
    - 3.2 “modify existing subject”.
    - 3.3 “modify existing course”.
    - 3.4 “modify existing term”.
    - 3.5 “modify existing calendar event”.
4. The system retrieves a list of all entity instances and displays it to the admin.
5. The admin picks an entity instance to modify.
6. The system retrieves the entity data of the selected instance and displays it to the admin in a modifiable form.
7. The admin makes changes to data fields in the form and submits.
8. The system updates the entity instance.



Communication Diagram of realization of Scenario 3

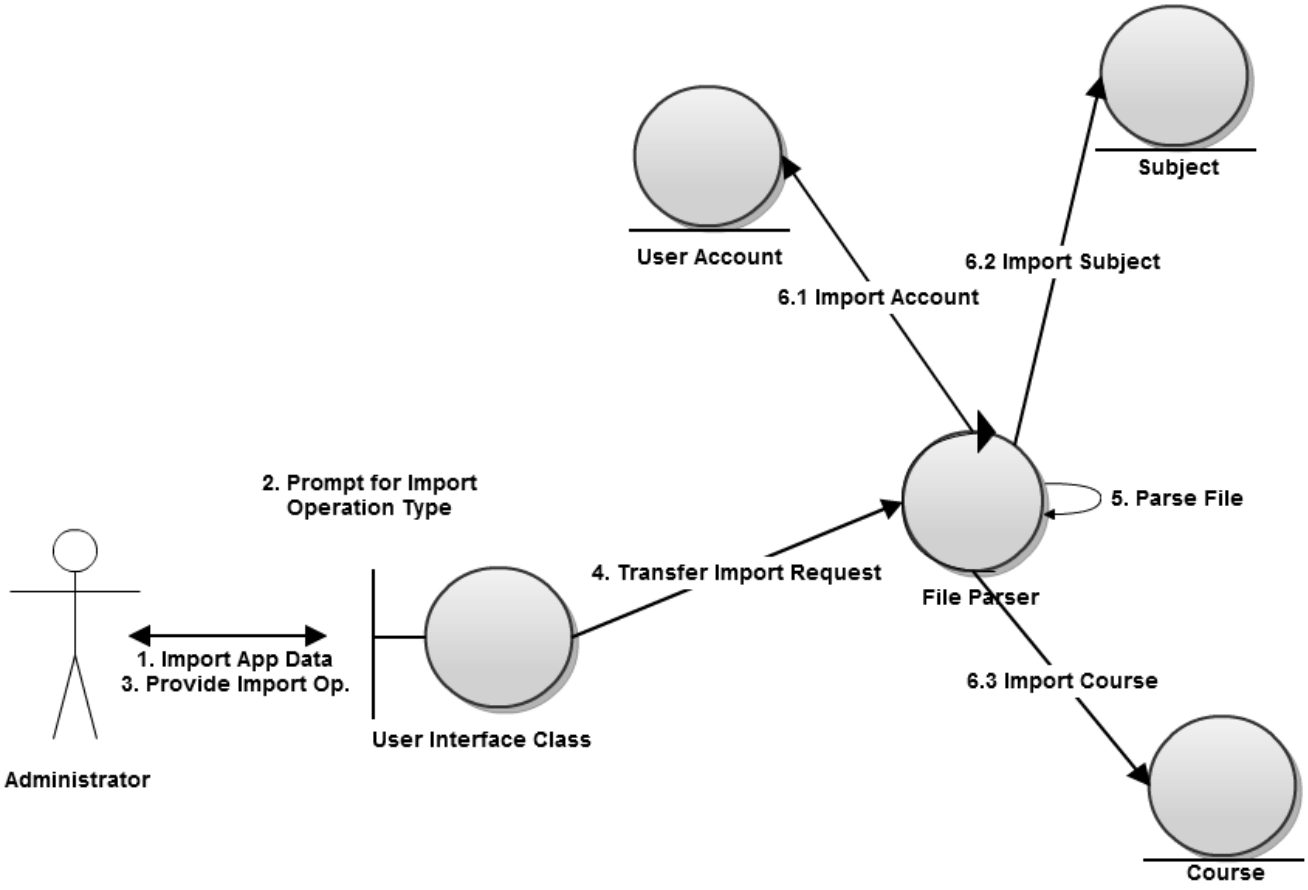
Scenario 4:

An administrator wishes to import data into the system.

1. The administrator navigates to import application data.
2. The system prompts the user to select one of the following options: import users, subjects, or courses
3. The administrator selects an option

3.1 “import users”.
3.2 “import subjects”.
3.3 “import courses”.

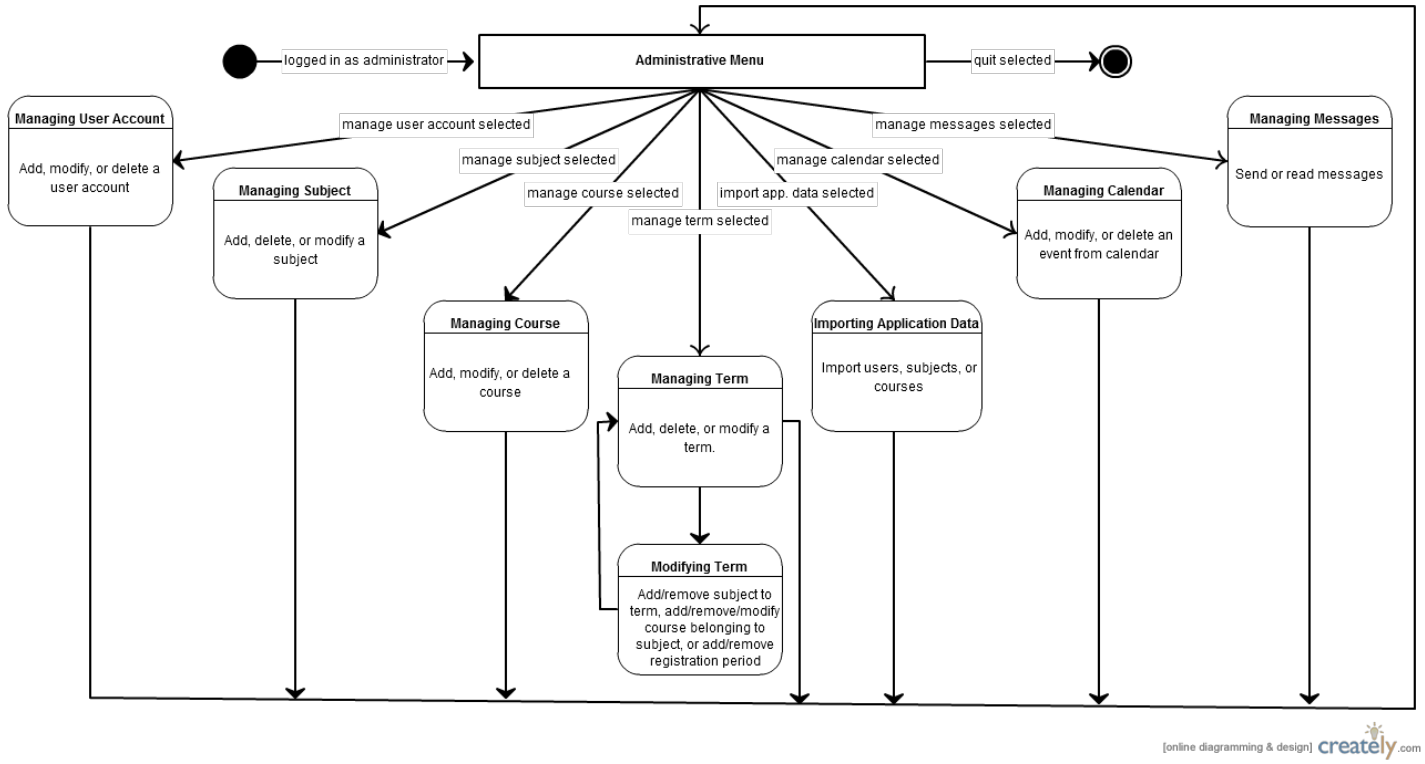
4. The system prompts for a file.
5. The administrator submits a file.
6. The system parses the file and imports the data.



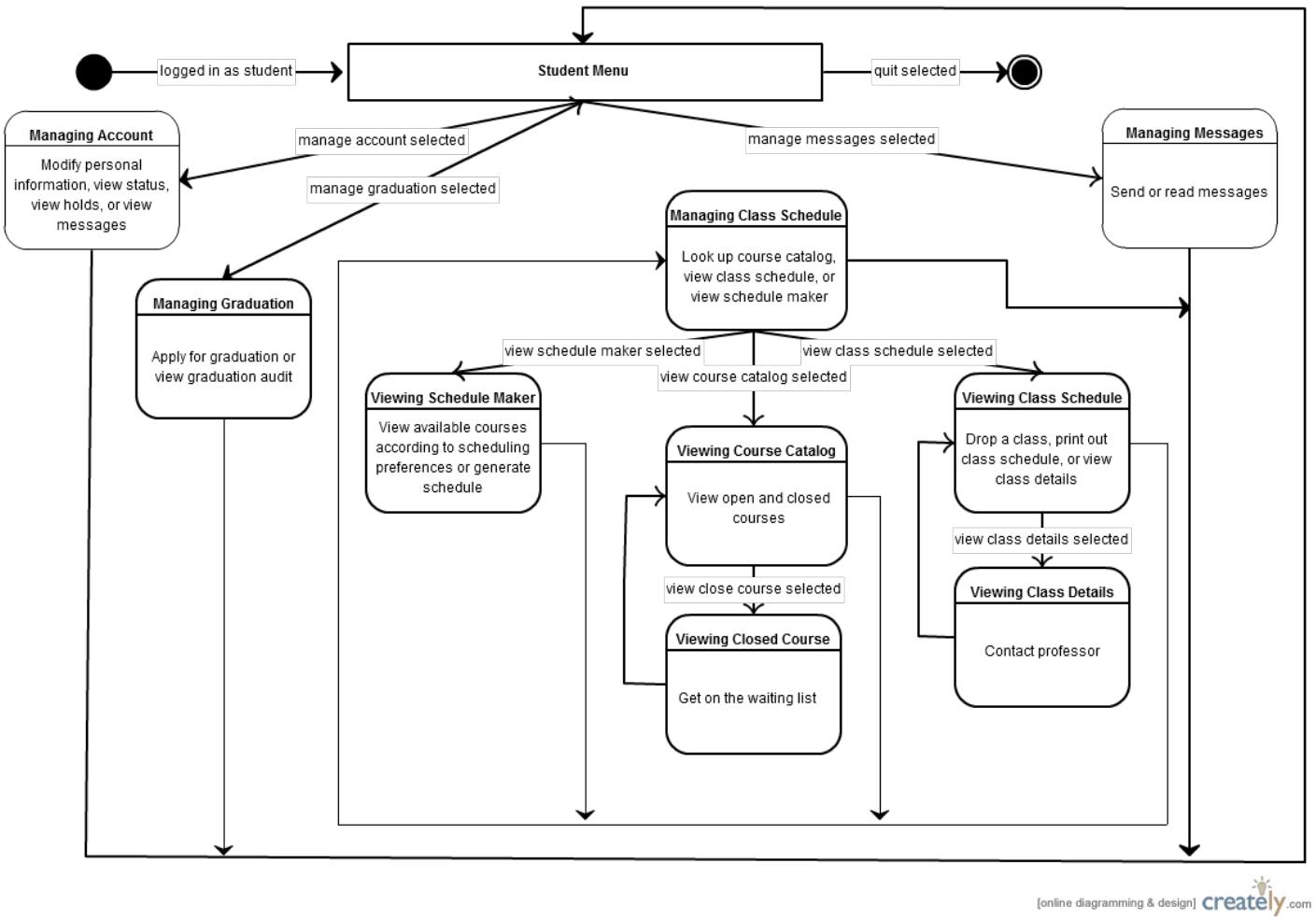
Communication Diagram of realization of Scenario 4

6. Dynamic Modeling

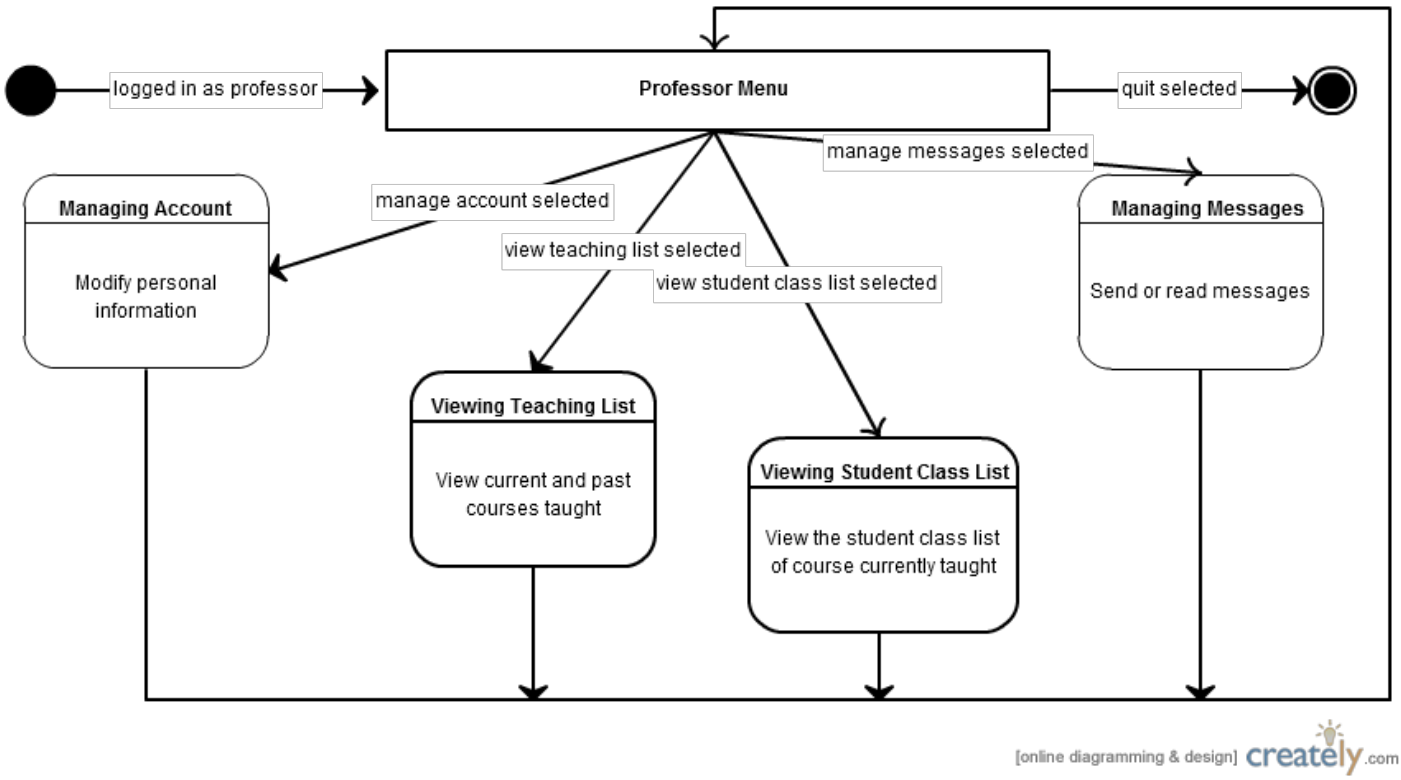
6.1 Admin


logged in as administrator
Administrative Menu
quit selected
Managing User Account
Add, modify, or delete a user account
manage user account selected
manage subject selected
manage course selected
manage term selected
import app. data selected
manage calendar selected
manage messages selected
Managing Subject
Add, delete, or modify a subject
Managing Course
Add, modify, or delete a course
Managing Term
Add, delete, or modify a term.
Modifying Term
Add/remove subject to term, add/remove/modify course belonging to subject, or add/remove registration period
Importing Application Data
Import users, subjects, or courses
Managing Calendar
Add, modify, or delete an event from calendar
Managing Messages
Send or read messages
[online diagramming & design] creately.com

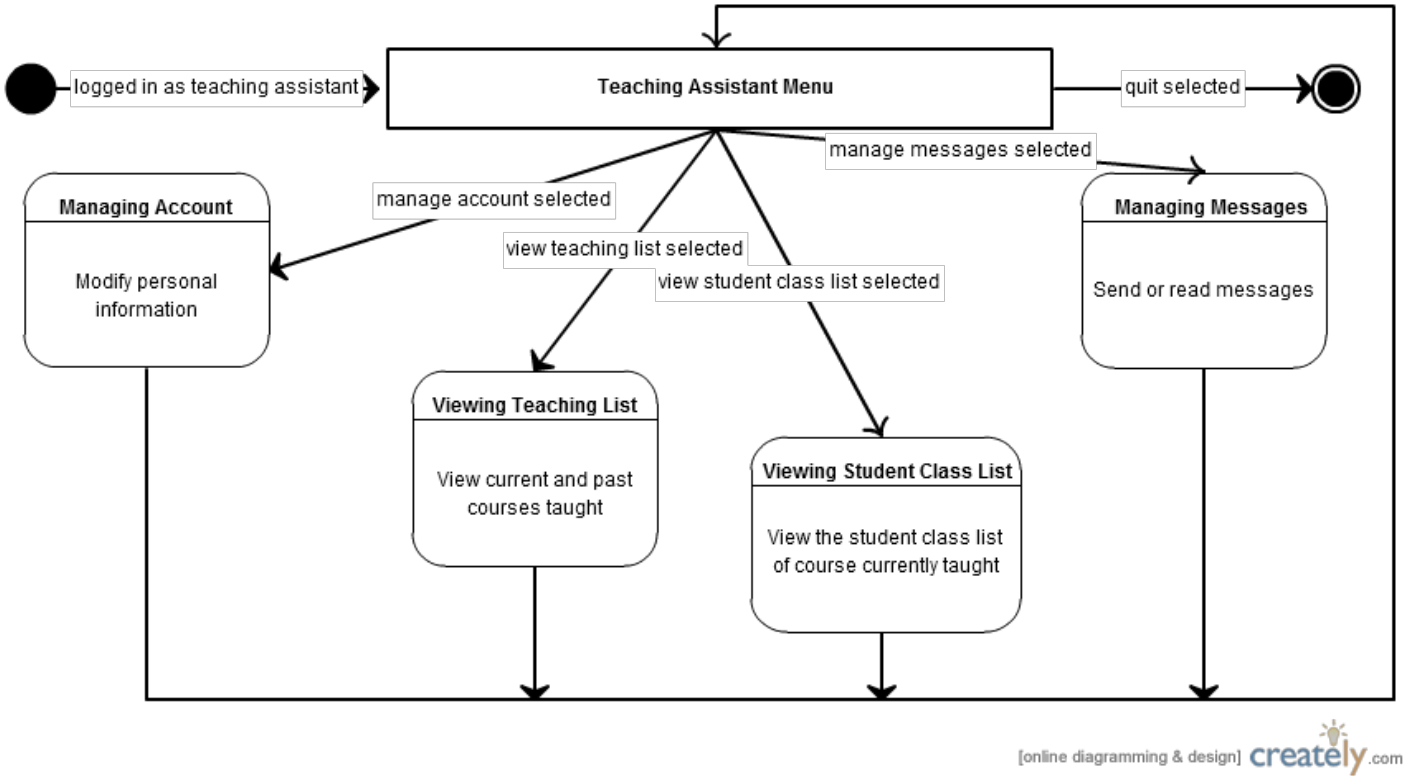
6.2 Student



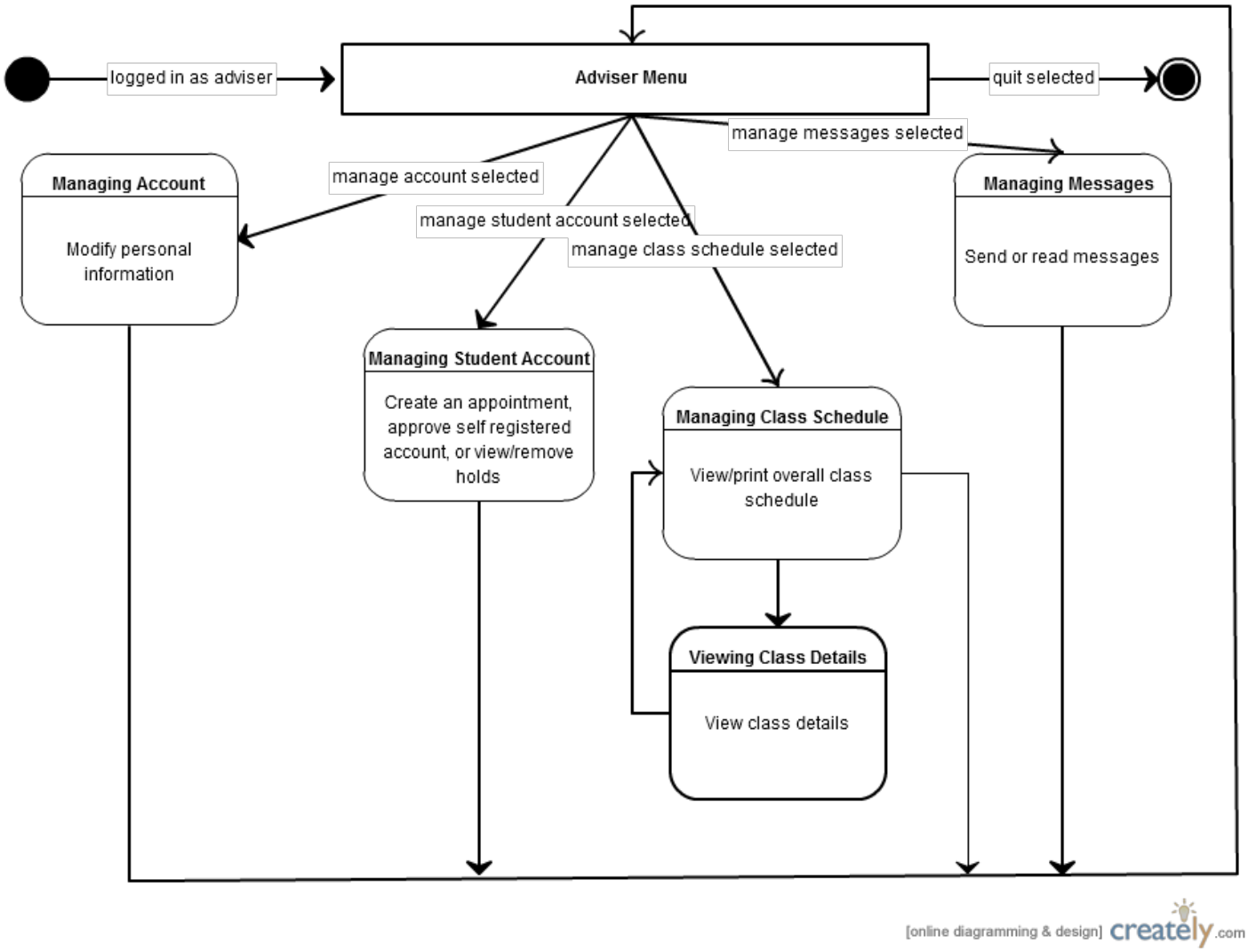
6.3 Professor



6.4 Teaching Assistant



6.5 Adviser



7. Performance Requirements

Non-Functional Requirements:

- Database should be normalized
- Application must be cross platform
    - Linux, Mac, Windows
- Response time must be under 3 secs for all operations
- Reliability
    - Database integrity for data types and required fields

- Must be able to import comma delimited txt files as well as xml files for initial database construction

8. Exception Handling

The system will have several types of possible exceptions. An invalid data exception will be thrown when user provided input fails validation check performed before insertion into the database. This exception throws an error and the use is notified to correct her mistake when the provided input type does not match the expected type. This exception will preserve the integrity of the application data. An invalid file exception will be thrown by the import application data operation when a file format is not recognized to be one of the allowable file formations (either CSV or XML standard). The file parser will also throw an exception, when there is a parse error, notifying the user of the problem location. Any data being imported that fails validation will not be inserted, and a warning will be exerted to the user, however, remaining data that passes validation will be inserted as usual.

9. Foreseeable Modifications and Enhancements

During the design phase a strong focus will be on presenting an intuitive and easy to use GUI for all users. In particular the way classes are searched and signed up for by the Student may change several times. One possible change is to model this procedure closer to an online shopping experience. The addition of more entity, control, or boundary classes and respective documentation may be needed.

10. Functional and Performance Test Plan

1. Data and Database Integrity Test

In this test, we will ensure that database is working properly. The objective will be to ensure that databases methods are proper and do not corrupt the data.

a. Each method will be called, and both valid and invalid data will be passed to ensure proper working.

b. Retrieval queries will be made to ensure that data is correctly populated.

2. Business Cycle Testing

This is to ensure that all background processes are working according to the functionality.

a. All the modules will be checked by passing valid and invalid data.

b. Check the output, whether it is correct or not, in case of valid data.

c. In case of valid data, it will be checked whether proper error messages are displayed to the user.

3. User Interface Testing

This is to ensure that all menus are working properly and all navigations are correct.

a. All the links / buttons on the page will be clicked one by one.
b. It will be checked whether any link is broken or not.
c. In case of not broken, it will be verified whether the page retrieved is the correct one or not.

4. Security Testing

This test will be used to ensure that a user without proper authentication should not be able to access the system. This test will also ensure that the user, who is a valid user, cannot access privileged functions.

a. Once the username and password are entered, it will be checked whether the user is valid or not
b. In case the user /password is invalid, a proper error message will be displayed.
c. It will be ensured that no user can access the system using the basic hacking techniques.
d. If the user is valid, it will be ensured that he is not able to access the restricted functionality.
e. It will be ensured that sensitive data is only made accessible to those who are supposed to manage it.
f. If a user can only view some information, and not edit it, it will be ensured that he does not have the rights to modify that table.

Misc

USE CASE GENERATOR CODE

Please enter your code for your use cases here:

Student:

Login
[Student]-(Login)
[Student]-(Logout)

Manage Class Schedule

[Student]-(Manage Class Schedule)
(Manage Class Schedule)>(Look Up Classes Offered This Semester)
(Look Up Classes Offered This Semester)<(Open Classes)
(Look Up Classes Offered This Semester)<(Closed Classes)
(Closed Classes)>(Get on a Waiting List for Class)
(Manage Class Schedule)>(View Overall Class Schedule)
(View Overall Class Schedule)>(Drop a Class)
(View Overall Class Schedule)>(Print Out Schedule of Classes)
(View Overall Class Schedule)>(View Class Detail)
(View Class Detail)>(Contact Professor)
[Professor]-(Contact Professor)
(Manage Class Schedule)>(Schedule Maker)
(Schedule Maker)>(Generate Schedule)
(Schedule Maker)>(Show Available Classes for Free Times in Student's Schedule)

Send Message

[Student]-(Message manager)
(Message manager)>(Send message to user)
(Message manager)>(Read received message)

Manage Graduation

[Student]-(Manage Graduation)
(Manage Graduation)>(View Graduation Audit)

(Manage Graduation)>(Apply for Graduation)

Manage Account

[Student]-(Manage Account)
(Manage Account)>(Modify Personal Information)
(Manage Account)>(View Status [Full Time/Part Time])
(Manage Account)>(View Holds)
(Manage Account)>(View Messages)

Professor
Login
[Professor]-(login)
[Professor]-(logout)

Manage Account
[Professor]-(Manage Account)
(Manage Account)>(Modify Personal Information)

Send / Receive Messages
[Professor]-(Message manager)
(Message manager)>(Send message to user)
(Message manager)>(Read received message)

Other Cases
[Professor]-(See the list of courses that he is teaching)
[Professor]-(See the list of students that are taking his course)
(See the list of courses that he is teaching)>(During Current Semester)
(See the list of courses that he is teaching)>(During Previous Semester)
(See the list of students that are taking his course)>(print the list of students for course)

Teacher Assistant
Login

[Teacher Assistant]-(login)
[Teacher Assistant]-(logout)

Send/Receive Messages
[Teacher Assistant]-(Message manager)
(Message manager)>(Send message to user)
(Message manager)>(Read received message)

Manage Account
[Teacher Assistant]-(Manage Account)
(Manage Account)>(Modify Personal Information)

Other Cases
[Teacher Assistant] - (see the list of courses that he is TA of)
[Teacher Assistant] - (see the list of students of the course he is TA of)
(see the list of courses that he is TA of)>(During Current Semester)
(see the list of courses that he is TA of)>(During Previous Semester)

Adviser:

Login
[Adviser]-(Login)
[Adviser]-(Logout)

Manage Class Schedule
[Adviser]-(Edit Class)
(Edit Class)>(Override Class Capacity)

Manage Account
[Adviser]-(Manage Account)
(Manage Account)>(Modify Personal Information)
(Manage Account)>(View Status [Full Time/Part Time])
(Manage Account)>(View Holds)
(Manage Account)>(Remove Holds)
(Manage Account)>(Approve self registration)
(Manage Account)>(Create appointment)
(Manage Account)>(Pick a Student)

Administrator:
[Administrator]-(Manage user account)
(Manage user account)>(Add new user account)
(Manage user account)>(Remove existing user account)

(Manage user account)>(Modify existing user account)
(Modify existing user account)>(Place hold on student account)
(Modify existing user account)>(Remove hold from student account)

[Administrator]-(Manage subject)
(Manage subject)>(Add new subject)
(Manage subject)>(Remove existing subject)
(Manage subject)>(Modify existing subject)

[Administrator]-(Manage course)
(Manage course)>(Add new course)
(Manage course)>(Remove existing course)
(Manage course)>(Modify existing course)

[Administrator]-(Manage term)
(Manage term)>(Add new term)
(Manage term)>(Remove existing term)
(Manage term)>(Modify existing term)
(Modify existing term)>(Add subject to term)
(Modify existing term)>(Remove subject from term)
(Modify existing term)>(Add course to subject)
(Modify existing term)>(Remove course from subject)
(Modify existing term)>(Modify course belonging to subject)
(Modify existing term)>(Add registration period)
(Modify existing term)>(Remove registration period)

[Administrator]-(Manage calendar)
(Manage calendar)>(Add new calendar event)
(Manage calendar)>(Remove existing calendar event)
(Manage calendar)>(Modify existing calendar event)

[Administrator]-(Message manager)
(Message manager)>(Send message to user)
(Message manager)>(Read received message)

[Administrator]-(Import application data)
(Import application data)<(Import users)
(Import application data)<(Import subjects)

(Import application data)<(Import courses)
(Import users)>(Parse file and insert data)
(Import subjects)>(Parse file and insert data)
(Import courses)>(Parse file and insert data)

Class diagrams:

[User Account
Class|Name:String;Surname:String;Address:String;Phone:Integer;Email:String;Id:Integer;R
ole:Administrator/Adviser/TA/Professor/Student]
[Academic Account Class|Emergency contact:String]
[Student Account Class|Role=Student;Major:String;GPA:Float;Adviser(ID):Integer;Enrolled
Courses:Set(Term Courses)]
[Professor Account Class|Role=Professor;Teaching:Set(Term Courses)]
[TA Account Class|Role=TA;Teaching:Set(Term Courses)]
[Advising Account Class|Role=Adviser;Advisees(IDs):Set(Integers)]
[Administrative Account Class|Role=Administrator]
[User Account Class]^[Academic Account Class], [Academic Account Class]^[Student
Account Class], [Academic Account Class]^[Professor Account Class], [Academic
Account Class]^[TA Account Class], [Academic Account Class]^[Advising Account Class],
[User Account Class]^[Administrative Account Class]

[Hold Class|Active:Boolean;PlacedBy(ID):Integer]
[Unpaid Tuition Hold Class|Balance:Integer]
[Medical Hold Class|Immunization:Boolean;Health Insurance:Boolean]
[Academic Standing Hold Class|Description:String]
[Advising Hold Class|Advisor(ID):Integer]
[Hold Class]^[Medical Hold Class], [Hold Class]^[Advising Hold Class], [Hold
Class]^[Unpaid Tuition Hold Class], [Hold Class]^[Academic Standing Hold Class]

[Subject Class|Name:String;College:String;Department:String]

[Course Class|Name:String;Subject:String;Description:String;Prerequisite
Requirements:String;Credit Hours:Integer]

[Term Subject Class|Term(ID):Integer;Set(Term Courses)]
[Term Course Class|Subject:Term Subject;Classroom location:String;Meeting
Time:String;Class capacity:Integer;Course(ID):

String;Instructor(ID):Integer;TA(ID):Integer;Students(IDs):Set(Integers)]
[Term Class|ID:String;Registration Deadline:Date;Subjects:Set(Term Subjects)]

[Schedule Class|Student: User Account Class; Courses Signed Up For: Term Course Class Array]

Entity Relationships

[Student Account Class]<1-m>[Hold Class]
[Advising Account Class]<1-m>[Student Account Class]
[Subject Class]<1-m>[Course Class]
[Term Subject Class]<1-m>[Term Course Class]
[Term Class]<1-m>[Term Subject Class]
[Term Course Class]<m-1>[Professor Account Class]
[Term Course Class]<m-1>[TA Account Class]
[Term Course Class]<1-m>[Student Account Class]
[Subject Class]^[Term Subject Class]
[Course Class]^[Term Course Class]

[Calendar Class|Date;Event]
[Date Class|Year: Integer; Month:Integer; Day: Integer; Format:String{mm/dd/yyyy}]
[Event Class|Name:String;Description:String;Event Time:Date]
[Calendar Class]++1-n>[Date Class]
[Calendar Class]++1-n >[Event Class]

[Schedule Maker Diagram Class|Course Name: String; Course Time: String;Weekday: String; Time: String;getSchedule();printSchedules()]

[Display Course Offerings Class| Course Name: String; Weekday: String; Course ID: String; printClassResults();printAvailClasses()]

[Display Overall Course Schedule Class| Role:String;Print Detail Schedule:boolean; printSchedule();
printClassDetailSchedule();printClassDetailSchedule();printListOfStudents()]

[Manage Account Display Class| Role:String;
displayAccountView(Role);DisplayManagementTools()]

[Manage Graduation Display Class|Student:String]

[Database Management Screen|generateDisplay();]

[Print Students List Class| studentsList:Array; printStudentsList(studentsList);]

[Login and Logout Class| username:string; password:string; login();saveChanges();]

[Calendar View Class| showMonthlyView(); showWeeklyView(); showDailyView()]

[Schedule Maker Class|From Scratch:boolean;Student Has Courses: boolean;Schedule: Schedule Class;Courses Interested In: String Array;Term Courses Interested In: Term Course Class Array;Possible Schedules: Schedule Class Array;Generate Week Schedules();Get Current Schedule();Add Class();Delete Class();]

[Course Offerings Class| inputDepartment();inputClassName();getClassInfo()]

[Calculate GPA Class| GPA: Float; calculateGPA()]

[Database Class|query:String;result:String;]

[Message Manager Class|Message:String;]

[Admin Tools Class| Role:String;manageTerm();manageCalendar();manageUser();manageSubject();manageCourse();manageAccount()]

[Get Students List for Course Class| courseId:string; getStudentsListForCourse(courseId)]

[Authentication Class| authenticateUser(); showErrorMessage();]

[AssignEventToCalander| event:Event; eventDate:Date; location: string;giveEventToCalendarAtTime();checkIfNoClash();AssignEventOnDate();]

calendar class at url http://www.yuml.me/diagram/scruffy/class/%5BCalendar%20Class%7CDate;Event%5D,%20%5BDate%20Class%7CYear:%20Integer;%20Month:Integer;%20Day:%20Integer;%20

Format:String%7Bmm/dd/yyyy%7D%5D,%20%5BEvent%20Class%7CName:String;Description:String;Event%20Time:Date%5D,%20%5BCalendar%20Class%5D++1-n%3E%5BDate%20Class%5D,%20%5BCalendar%20Class%5D++1-n%20%3E%5BEvent%20Class%5D.pdf