Drop Grade

Software Requirement Specifications

Developed by: Alan Donham, Madison McHam, Shane Peters, and Nick Levert

|  |  |
| --- | --- |
| Table of Contents |  |
| Introduction | 3 |
| Description | 4 |
| Functional Requirements | 5 |
| Class Diagram | 6 |
| Activity Diagrams | 7 |
| Use Case Diagrams | 16 |
|  |  |

**Change History**

* 9/14/15 – Creation of document
* 9/16/15 – Add project Description – Shane Peters
* 9/16/15 – Add introduction – Alan Donham
* 9/16/15 – Add Functional and Nonfunctional Requirements – Madison McHam
* 9/16/15 – Add use case diagrams – Nick Levert
* 9/17/15 – Changed formatting to improve overall look – Alan Donham
* 9/17/15 – Added class diagram – Madison McHam
* 9/18/15 – Added activity diagrams – Shane Peters and Nick Levert
* 9/21/15 – Changed wording to improve readability - Group
* 10/1/15 – Added updated use case diagram - Group
* 10/1/15 – Updated to accommodate faults – Group
* 11/1/15 – Added detailed class diagram – Alan Donham

1. **Introduction**

This document will provide all the necessary information on the requirements necessary for development of the DropGrade application.

* 1. **– Purpose**

The purpose of this app is to provide the user not only with a way of tracking their course average in a class, but also with detailed information about professors and classes they are interested in registering for. Many apps offer features similar to this, however there does not exist one currently that is able to bundle these features we are offering into one sleek, easy to use, yet incredibly informative application like ours will.

* 1. **- Goals and Challenges**

One of the biggest challenges this application will face will be gathering and storing all the data collected in a way that will allow us to provide useful statistics to the user very quickly and in real time. This biggest challenge is also our number one goal because it is important that the user is able to acquire the data they want in near real time or they will likely not use the application we develop. Overall our goal is to provide the user with an application that will allow them to calculate their grade and view what if scenarios depending on the possible grades they can make on upcoming assignments and exams.

* 1. **– Definitions**

What-If-View – A view that will provide the user with tools to manipulate their final course average based on predicted future scores

Course-View – A view that will list all the courses a student is currently enrolled in

Professor-Stats – A view that will list statistics on professors from data that is collected

Course-Edit-View – A view that will allow user to input information regarding a course

GPS (Grade Prediction System) – The subsystem responsible for calculating course averages

GPA – A scale used to represent an average score that ranges from 0 to 4.3

Required - A requirement listed as required is seen as a requirement for completion

Stretch – A requirement listed as stretch will be completed should time allow

* 1. **– Actors**
* Students at The University of Alabama (User)
* Server (Database)

1. **Description**

**2.1 – User Interface**

The interface for the app will be a basic touchscreen interface similar to most apps on the market. The app will open onto a home screen that will have a list of classes that the user has entered. The information will be presented to and gathered from the user by a combination of table views, text boxes, drop down menus, and keypad entry.

* 1. **– Storage**

Information about course instructors is sent to and stored on a database.

* 1. **– Communication**

Requires cellular data or an Internet connection to send/receive data from a remote server.

* 1. **- Application Functions**

This application is intended to serve as a grade calculator and teacher-rating tool. Users will be asked to enter a class and it’s syllabus information. As the users progress through the school term they will input their grades into the app and track their overall course grade. At the conclusion of the course the user will be asked to rate and give a description of their course instructor(s).

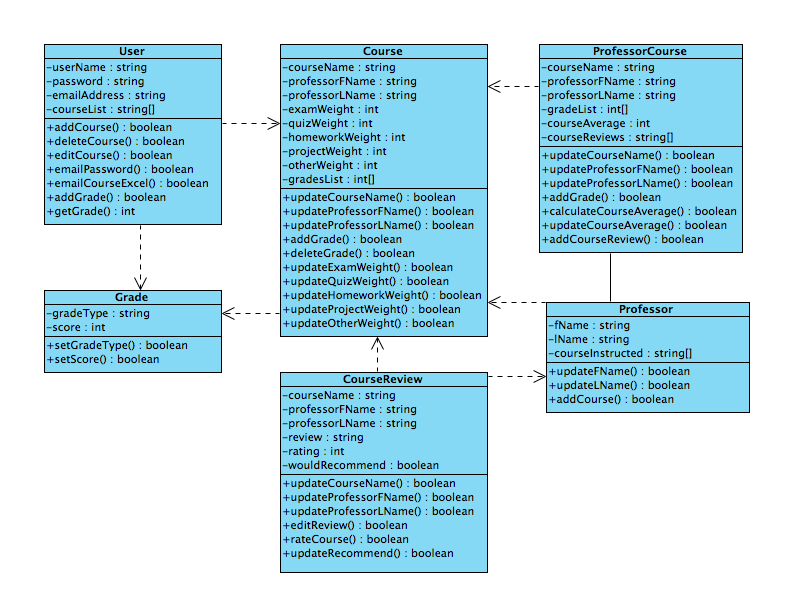
* 1. **- Assumptions**
* Grades will be given in a number format. User reported their grades honestly.
* User has internet access
  1. **- Stretch Goals**
* Reminder function that reminds user of a test or assignment.
* Have list of classes and their respective syllabi stored on server for easy download.
* Export course grade information to an excel spreadsheet

1. **Functional and Nonfunctional Requirements**

|  |  |
| --- | --- |
| Functional Requirement | Status |
| Create Account | Required |
| Log In | Required |
| Retrieve Password | Required |
| Add Course | Required |
| Add Course Info | Required |
| Enter Grades | Required |
| Edit Grade | Required |
| Submit Final Grade | Required |
| Manipulate Final Grade via the What-If-View | Required |
| Add Professor Review | Required |
| Delete Professor Review | Required |
| Edit Professor Review | Required |
| Export Data to an Excel Spreadsheet | Stretch |
| Send reminders to user when they have an upcoming Exam | Stretch |
| Store course information(including grading scale) | Stretch |

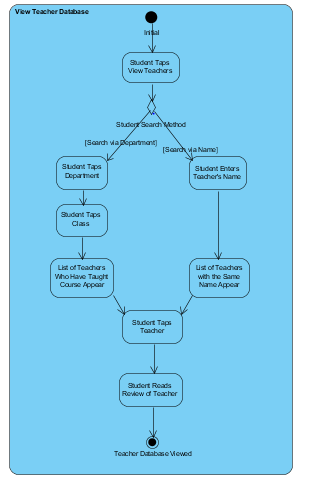
|  |  |
| --- | --- |
| Nonfunctional Requirement | Status |
| Database that can handle multiple users | Required |
| App must not drain battery | Required |

1. **Class Diagram**

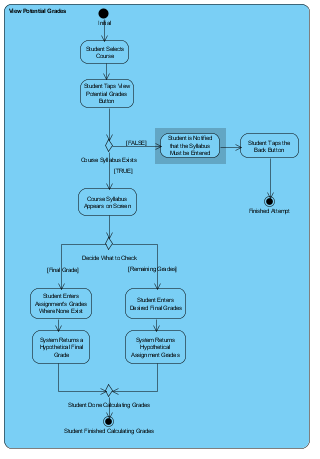
****

Our application will function in accordance with this diagram of our classes. As can be seen by looking at the diagram, everything is dependent on the User class which will store the necessary data to drive how our application talks to our database. The other classes listed above provide an organized way to store information, access it, and perform calculations on the information with ease.

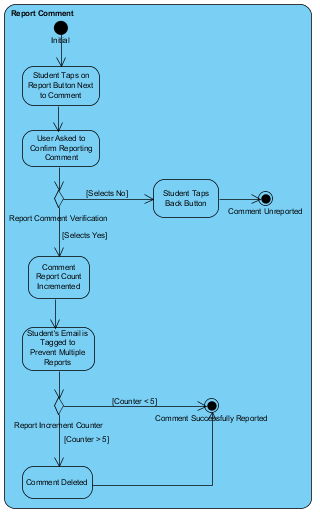
1. **Activity Diagrams**



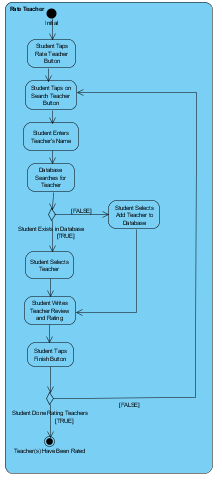
The user is able to access and read reviews about teachers.



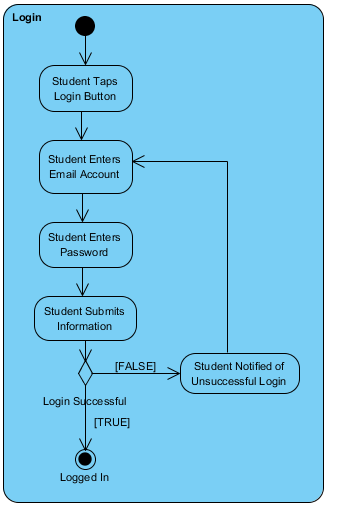
Allows student to see what grades they can potentially make based on his or her performance.



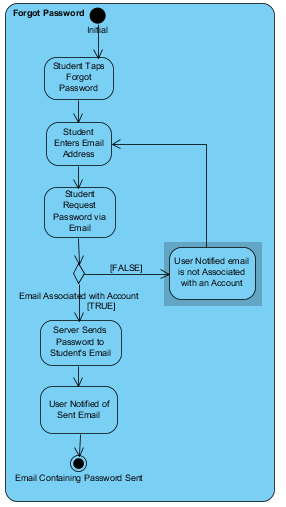
Allows user to report an incorrect or inappropriate review of a teacher.



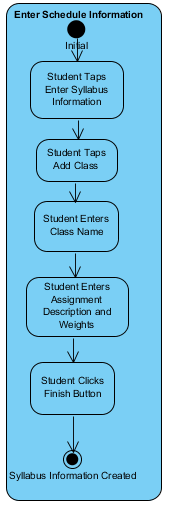
Allows user to rate their professor and add a review of them.



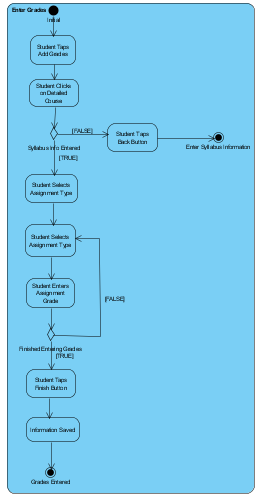
Allow user to login to app to access the app.



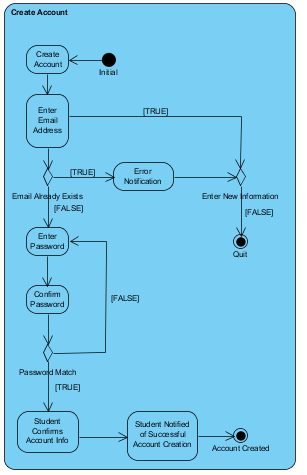
Allow user to reset their password if they have forgotten it.



Allows user to enter their classes into their schedule.



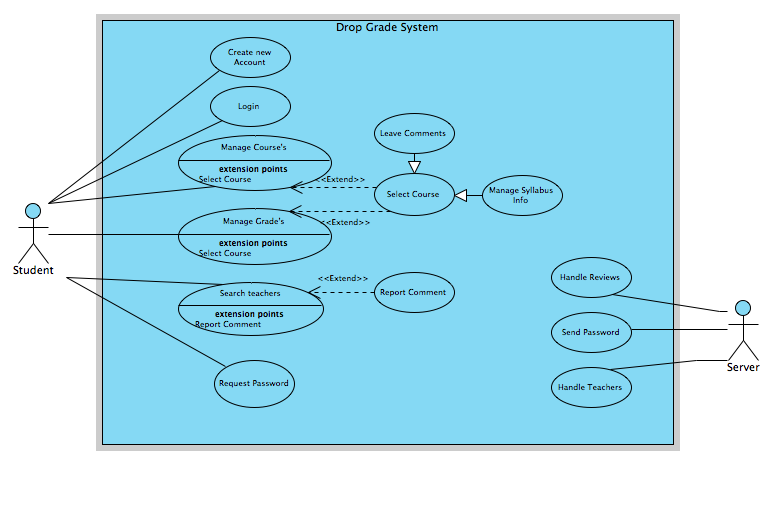
Allows user to enter grades for a class to calculate their grade.



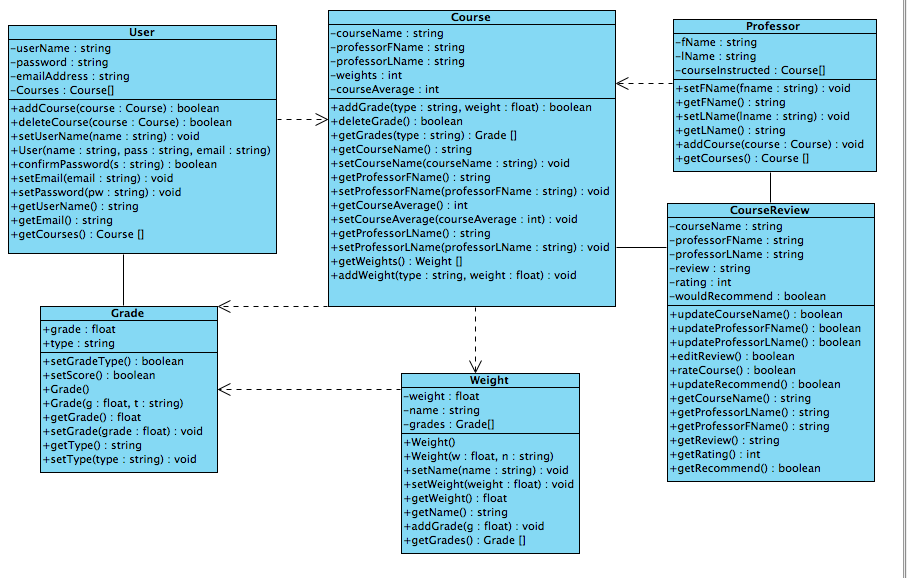
Allows user to create account to begin using the app.

1. **Use Case Diagrams**

Below are the use cases that our application will be designed from. The design of our classes branched from these use cases. More use cases may be added in the future, however for now it has been decided that these use cases will be what we are focused on satisfying with our design. This will help to focus the team and enforce quality time management. If time allows more use cases may be added to describe the stretch goals listed above.

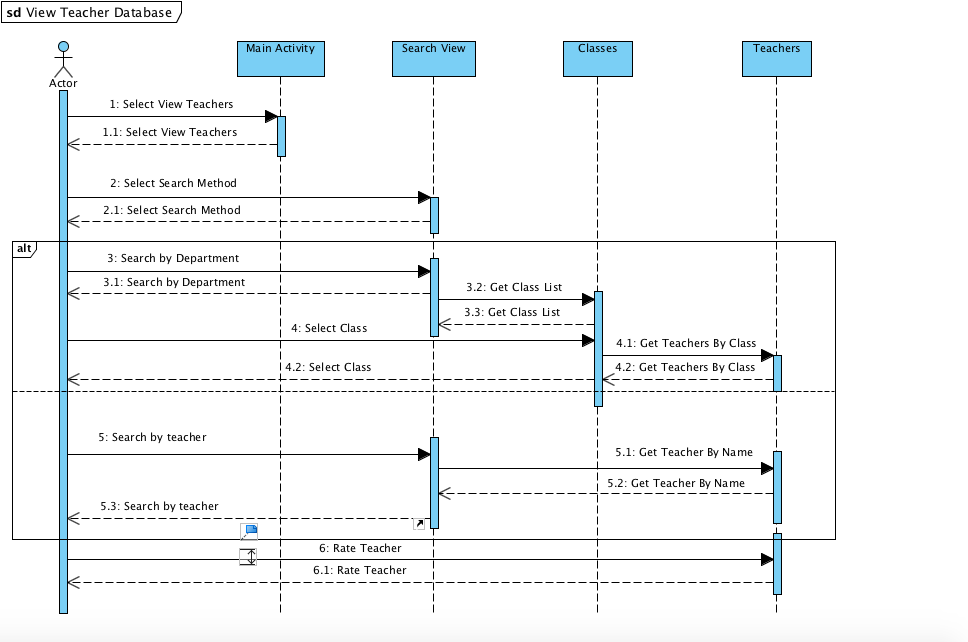


1. **Detailed Class Diagram**

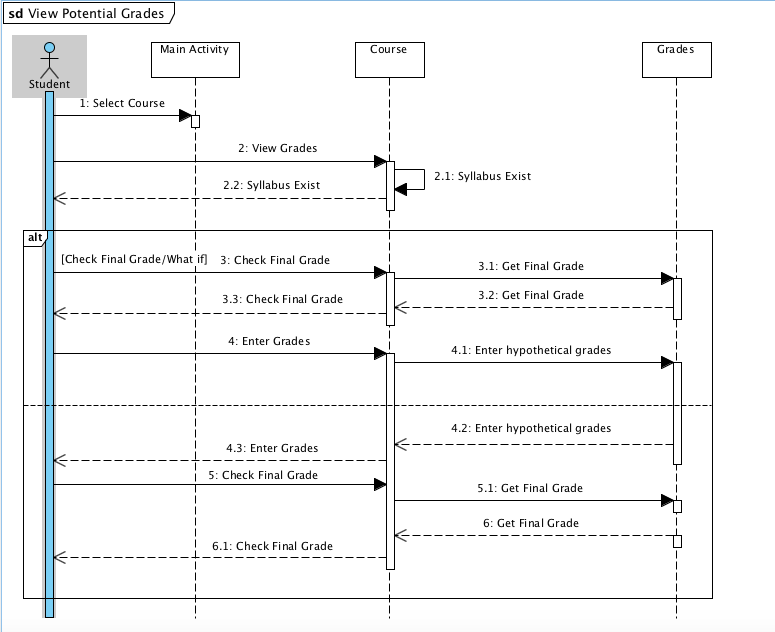


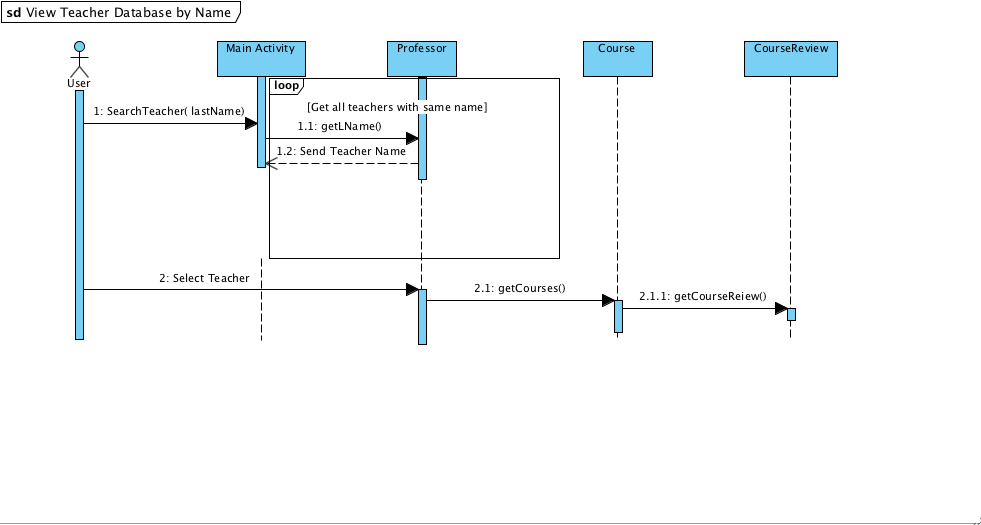
This class diagram outlines the structure of our classes for our application. All of these classes store data that is important to the app. The way the classes are structured, our application stores all of the information under the user class, which will allow us to integrate this into our database seamlessly by using the user ID as a means of uniquely accessing all the data needed. These classes all will also have the functionality needed to manipulate, get, and set the data stored in order to allow our application to work. Along with this, the classes will also perform the calculations required to provide our users with the what if situations and their class averages. In order to save storage space and reduce database complexity, we will perform each calculation in real time rather than storing the calculated values for later use. This should not affect performance though since the calculations required should be relatively simple and since the amount of calculations being performed will be relatively small. We are confident in this design and believe that it will allow our code to be very readable as well as easy to manage and efficient.

1. **Sequence Diagram**

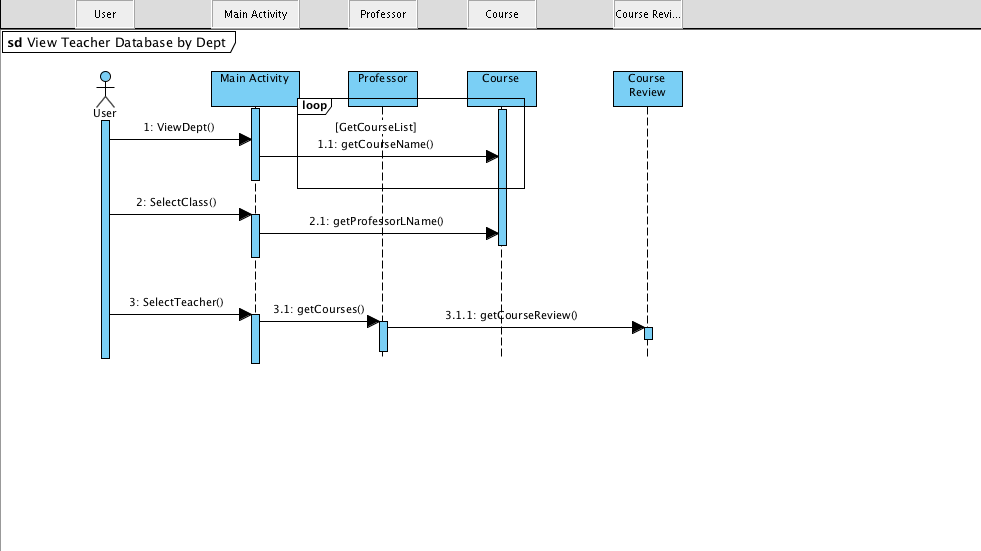
****

This diagram illustrates the use case where a student is looking up a professor in the database and then chooses to rate the professor. There are many categories available to filter the teachers.

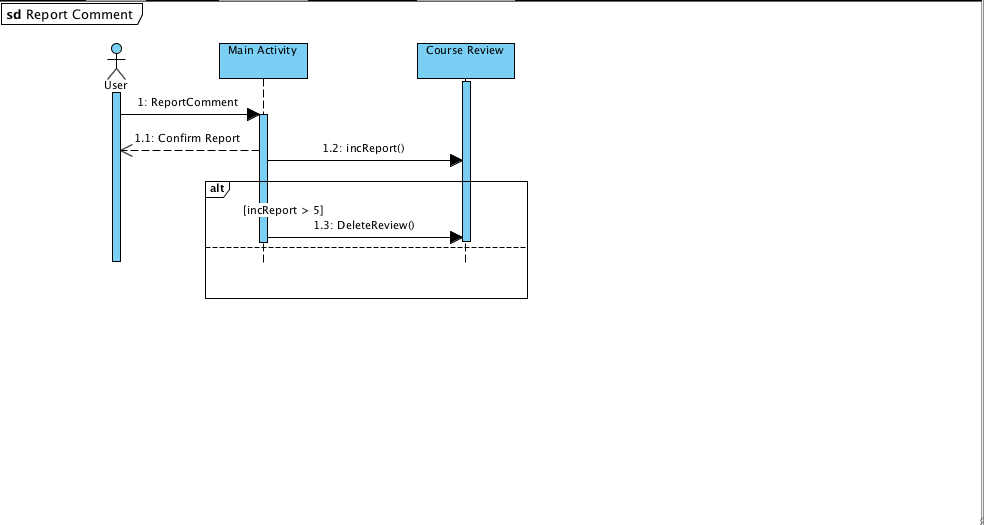
This diagram illustrates the use case where a student uses the application to generate what-if scenarios to predict their final grade in a course.

****

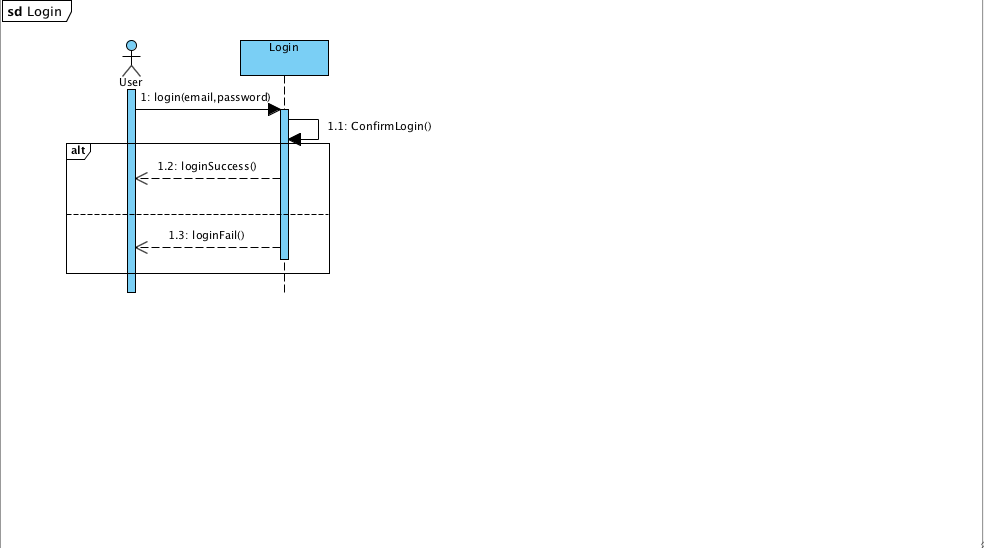
This diagram illustrates the use case where a student looks up a professor by their last name, chooses the professor they are looking for, and then is able to view their reviews.



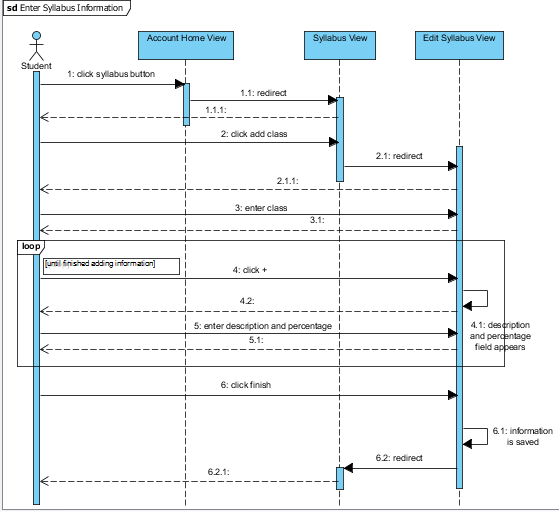
This diagram illustrates the use can where a student selects a professor from the database by searching the department the professor is a member of. They are then able to view the professor’s review.



This diagram illustrates the use case where an incorrect or inappropriate review comment has been created. The user is able to report the review and if it receives enough reports it will be deleted.



Tis diagram illustrates the login process for our application. It will be basic but functional.

****

This diagram illustrates how the user will edit the syllabus information per course. They will be able to enter the information about the class for each syllabus so it can be saved to the database and reused by other users.