Drop Grade

Software Requirement Specifications

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

TABLE OF CONTENTS

Introduction **4**

Description **5**

Functional Requirements **6**

Use Case Diagram **7**

Class Diagram **8**

Activity Diagrams **9**

View Teacher Database *9*
View Potential Grade *10*
Report Comment *11*
Rate Teacher *12*
Login *13*
Forgot Password *14*
Enter Schedule Information *15*
Enter Grades *16*
Create Account *17*

Detailed Class Diagram **18**

Sequence Diagrams **19**

View Teacher Database By Name *19*
View Teacher Database By Dept *19*
Report Comment *20*
Login *21*
Enter Syllabus Information *22*
View Potential Grades *23*
Forgot Password *23*
Rate Teacher *24*
Add Grades *25*
Create Account *26*

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
- 11/2/15 – Added sequence diagrams – Shane Peters and Nick Levert
- 11/3/15 – Added more descriptions to the Activity Diagrams- Madison McHam
- 11/3/15 – Put the Use Case diagram as the first diagram to appear – Madison McHam
- 11/3/15 – Added requirement descriptions – Alan Donham

1. INTRODUCTION

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

1.1 – 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.

1.2 - 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.

1.3 – 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

1.4 – Actors

- Students at The University of Alabama (User)
- Server (Database)

2. 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.

2.2 – Storage

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

2.3 – Communication

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

2.4 - 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).

2.5 - Assumptions

- Grades will be given in a number format. User reported their grades honestly.
- User has internet access

2.6 - 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

2.7 - Actors

- Students at The University of Alabama (User)
- Server (Database)

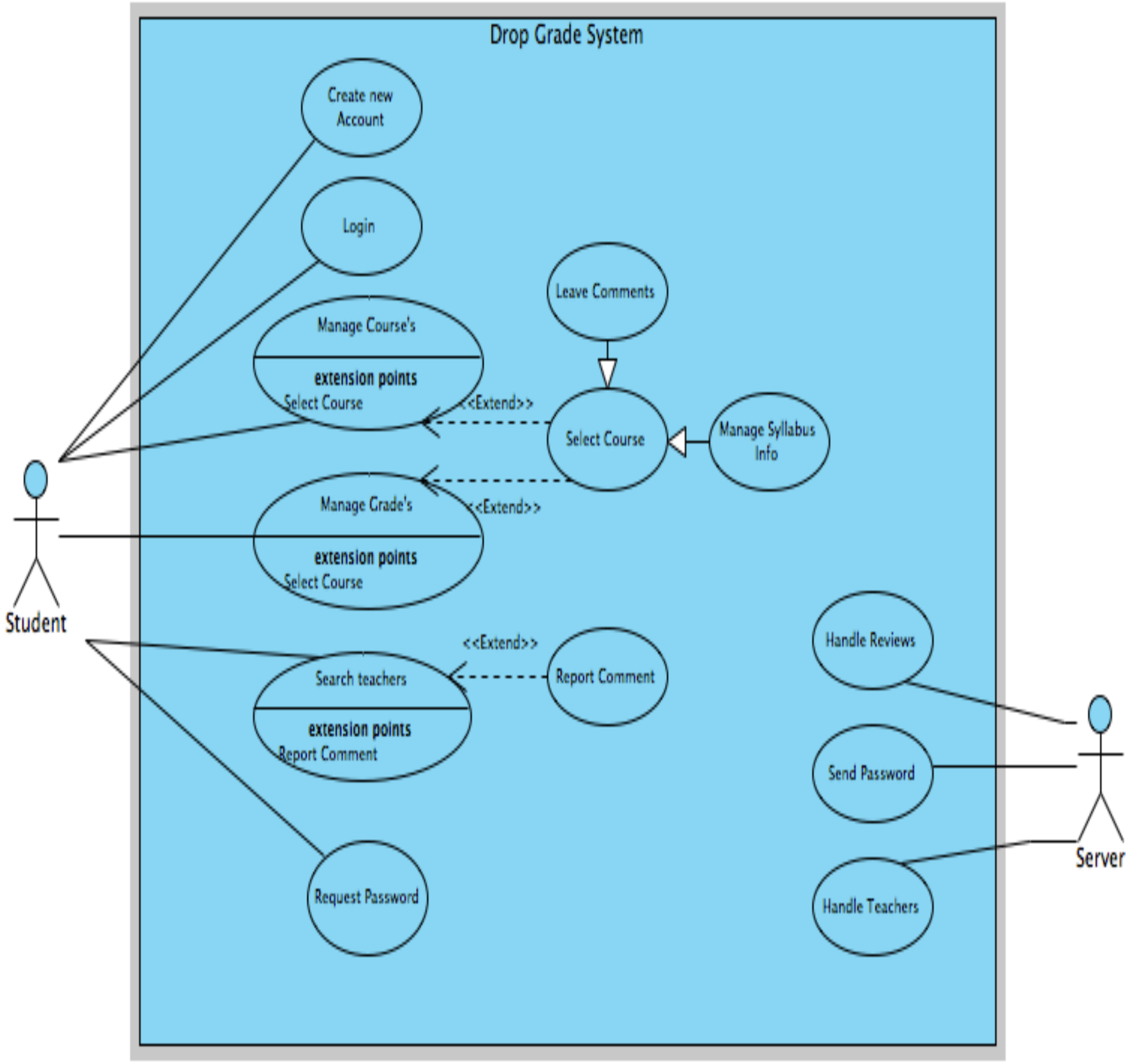
3. FUNCTIONAL AND NONFUNCTIONAL REQUIREMENTS

Functional Requirement	Status	Description
Create Account	Required	The user can create an account to login to the DropGrade App
Log In	Required	The user can use their account to login to the DropGrade App
Retrieve Password	Required	The user can retrieve a lost password
Add Course	Required	The user can add a course to their account
Add Course Info	Required	The user can update the course info on their account
Enter Grades	Required	The user can enter grades per course on their account
Edit Grade	Required	The user can edit/delete the grades per course on their account
Submit Final Grade	Required	The user can submit their final grade in a course to the server to be used for data analysis on professors
Manipulate Final Grade via the What-If-View	Required	The user can generate what-if scenarios to determine possible grade outcomes in the class
Add Professor Review	Required	The user can submit a review of a professor
Delete Professor Review	Required	The user can delete a review of a professor
Edit Professor Review	Required	The user can edit the review of a professor
Export Data to an Excel Spreadsheet	Stretch	The user can export their grades in a course to excel
Send reminders to user when they have an upcoming Exam	Stretch	The user will be sent reminders of when they have exams if they set this feature up
Store course information(including grading scale)	Stretch	The system will save course info for reuse by other students in the same class.

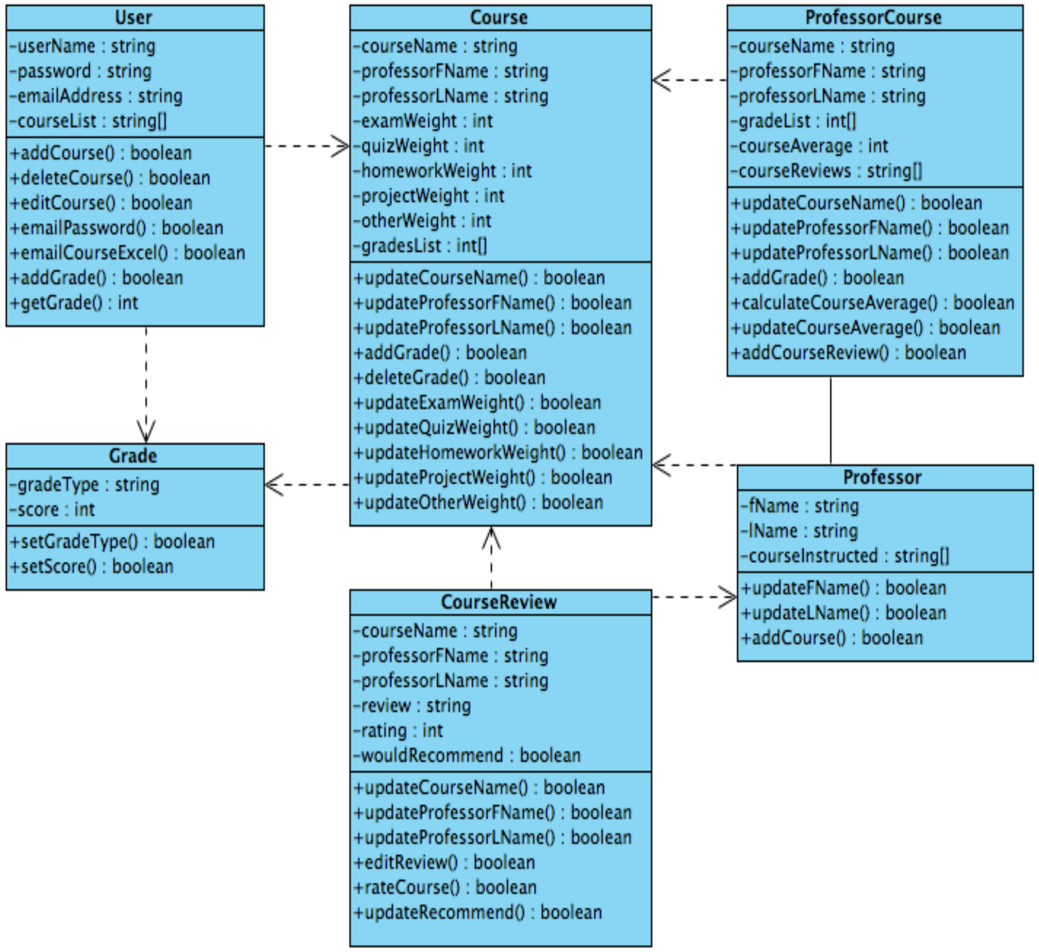
Nonfunctional Requirement	Status
Database that can handle multiple users	Required
App must not drain battery	Required

4. 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.



5. CLASS DIAGRAM

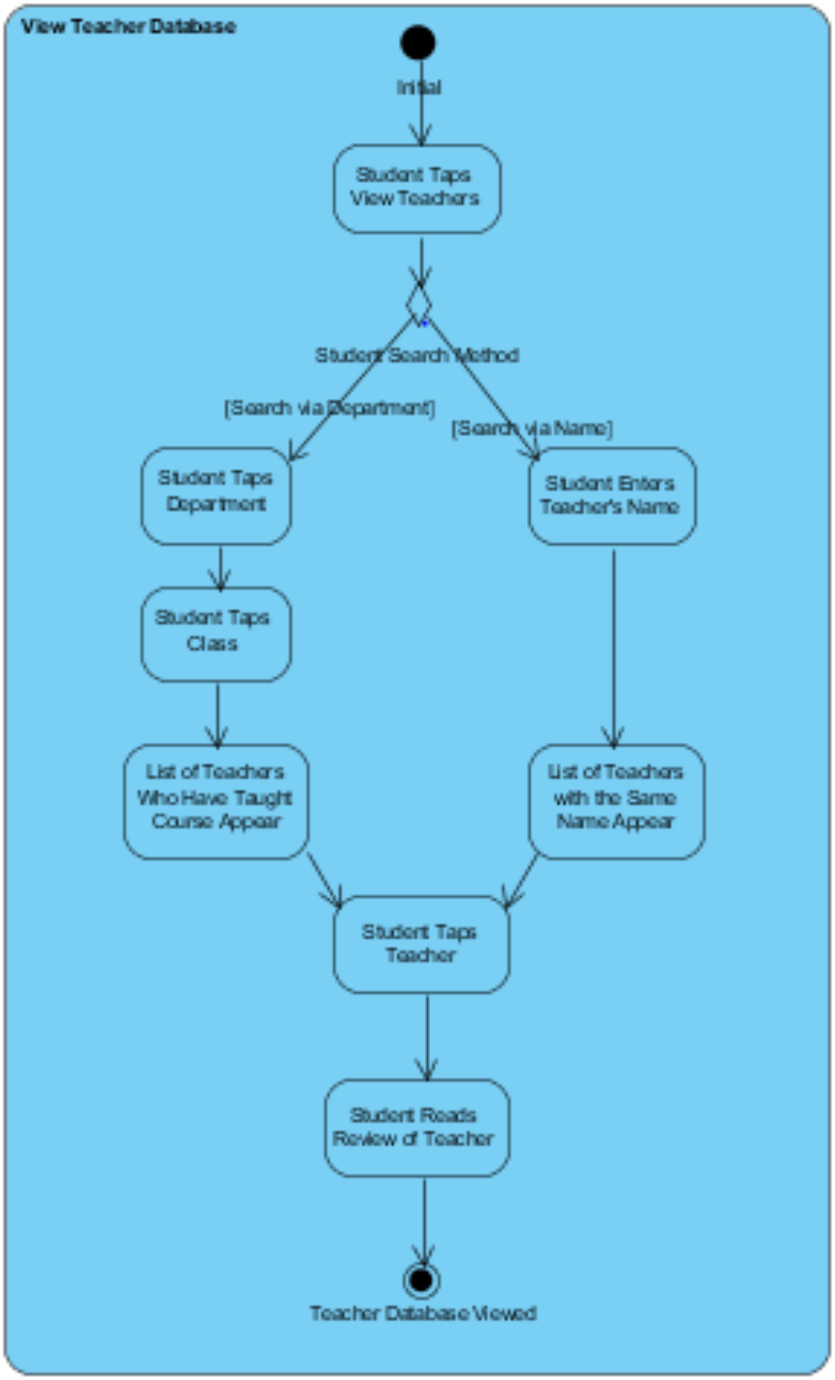


Our application will function in accordance with this diagram of our classes. As can be seen by looking at the diagram, everything is dependent on the User class, which will store the necessary data to drive how our application talks to our database. The other classes listed above provide an organized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

6. ACTIVITY DIAGRAMS

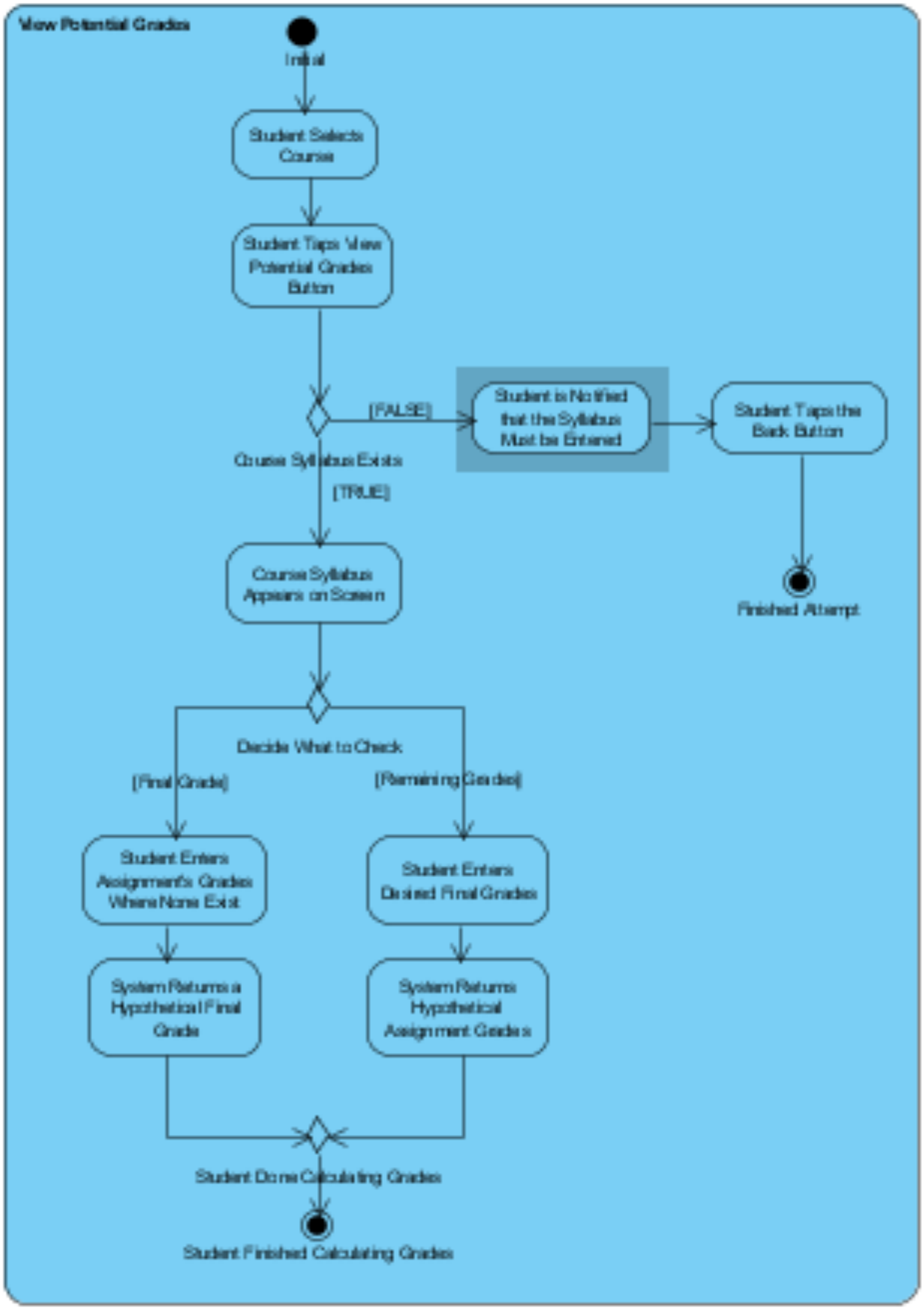
6.1 View Teacher Database

The user is able to tap on View Teachers and is faced with two choices. The user can enter a teachers name and a List of Teachers with that name appear or the user tap on Department tab then Class tab and find the wanted class and search threw teachers who have taught the class. Then they can access the Reviews for the chosen teacher.


View Teacher Database
Initial
Student Taps View Teachers
Student Search Method
[Search via Department]
[Search via Name]
Student Taps Department
Student Enters Teacher's Name
Student Taps Class
List of Teachers Who Have Taught Course Appear
List of Teachers with the Same Name Appear
Student Taps Teacher
Student Reads Review of Teacher
Teacher Database Viewed

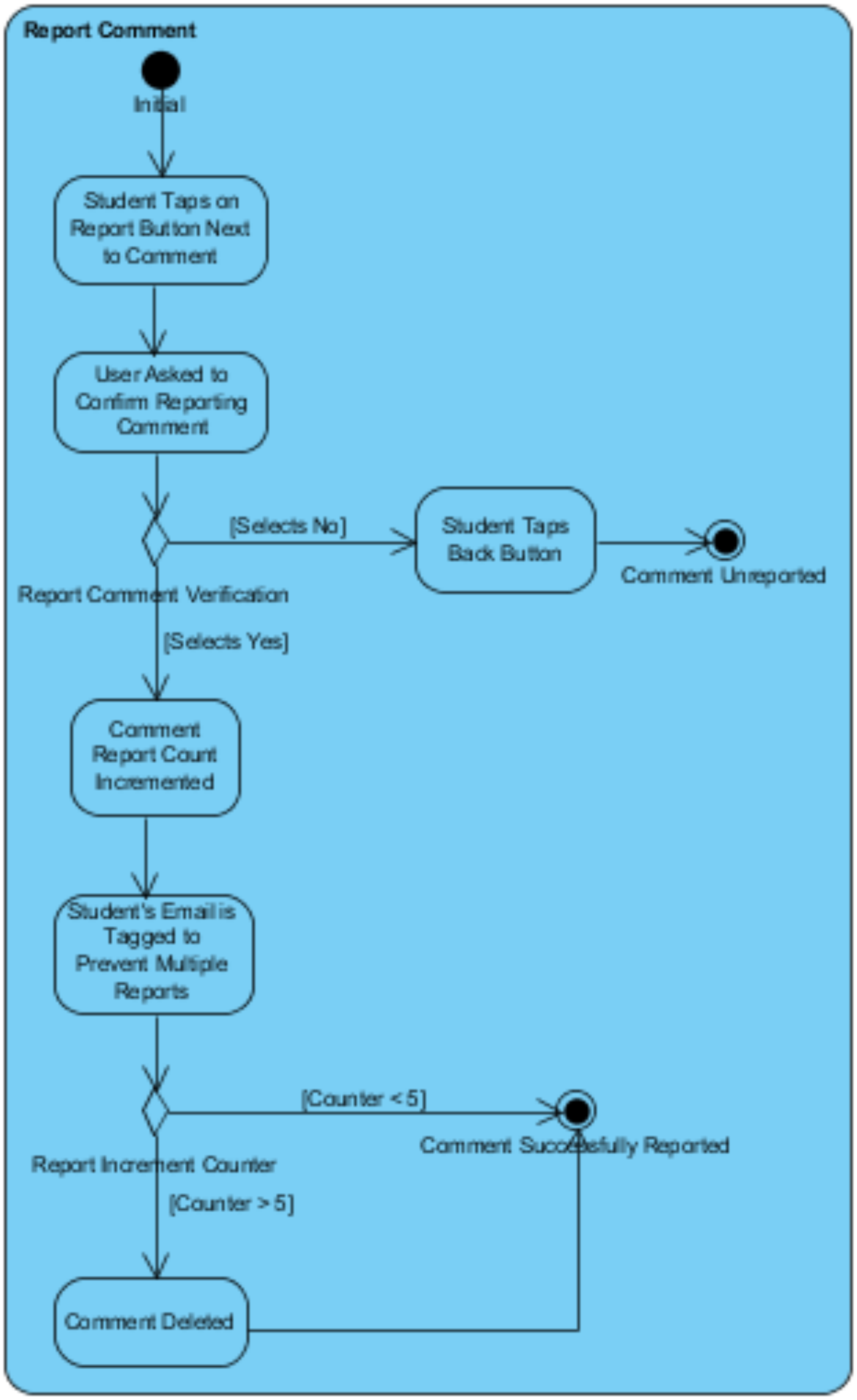
6.2 View Potential Grades

The user selects the Course he/she would like to figure out a grade for. Then the user taps View Potential Grades button and that takes user to another page and if the course syllabus does not exist the user will have to enter the percentages and grades manually. If the syllabus does exist the student then can play around with percentages and enter the grades for the workload. Allows student to see what grades they can potentially make based on his or her performance.


View Potential Grades
Initial
Student Selects Course
Student Taps View Potential Grades Button
[FALSE]
Student is Notified that the Syllabus Must be Entered
Student Taps the Back Button
Finished Attempt
Course Syllabus Exists
[TRUE]
Course Syllabus Appears on Screen
Decide What to Check
[Final Grade]
[Remaining Grades]
Student Enters Assignment's Grades Where None Exist
Student Enters Desired Final Grades
System Returns a Hypothetical Final Grade
System Returns Hypothetical Assignment Grades
Student Done Calculating Grades
Student Finished Calculating Grades

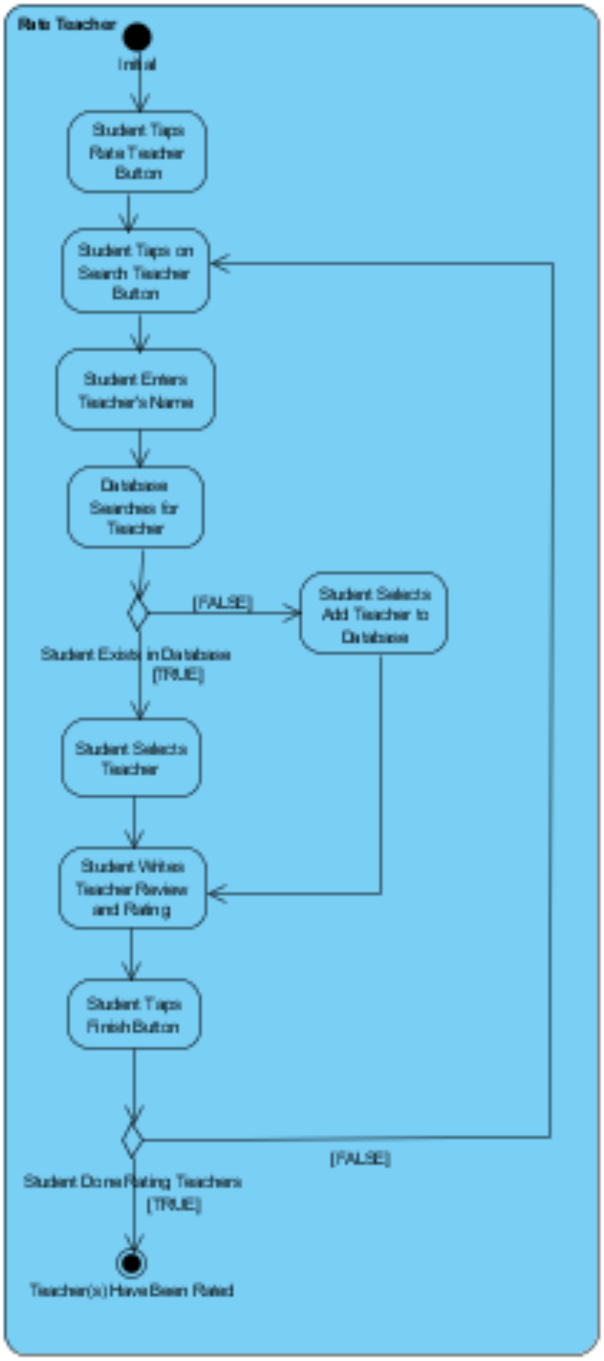
6.3 Report Comment

Allows user to report an incorrect or inappropriate review of a teacher. The report is sent to server which checks the amount of times that this review has been reported and if its less then 5 the comment stays and once it is reported over 5 times it will be removed.


Report Comment
Initial
Student Taps on Report Button Next to Comment
User Asked to Confirm Reporting Comment
[Selects No]
Student Taps Back Button
Comment Unreported
Report Comment Verification
[Selects Yes]
Comment Report Count Incremented
Student's Email is Tagged to Prevent Multiple Reports
[Counter < 5]
Comment Successfully Reported
Report Increment Counter
[Counter > 5]
Comment Deleted

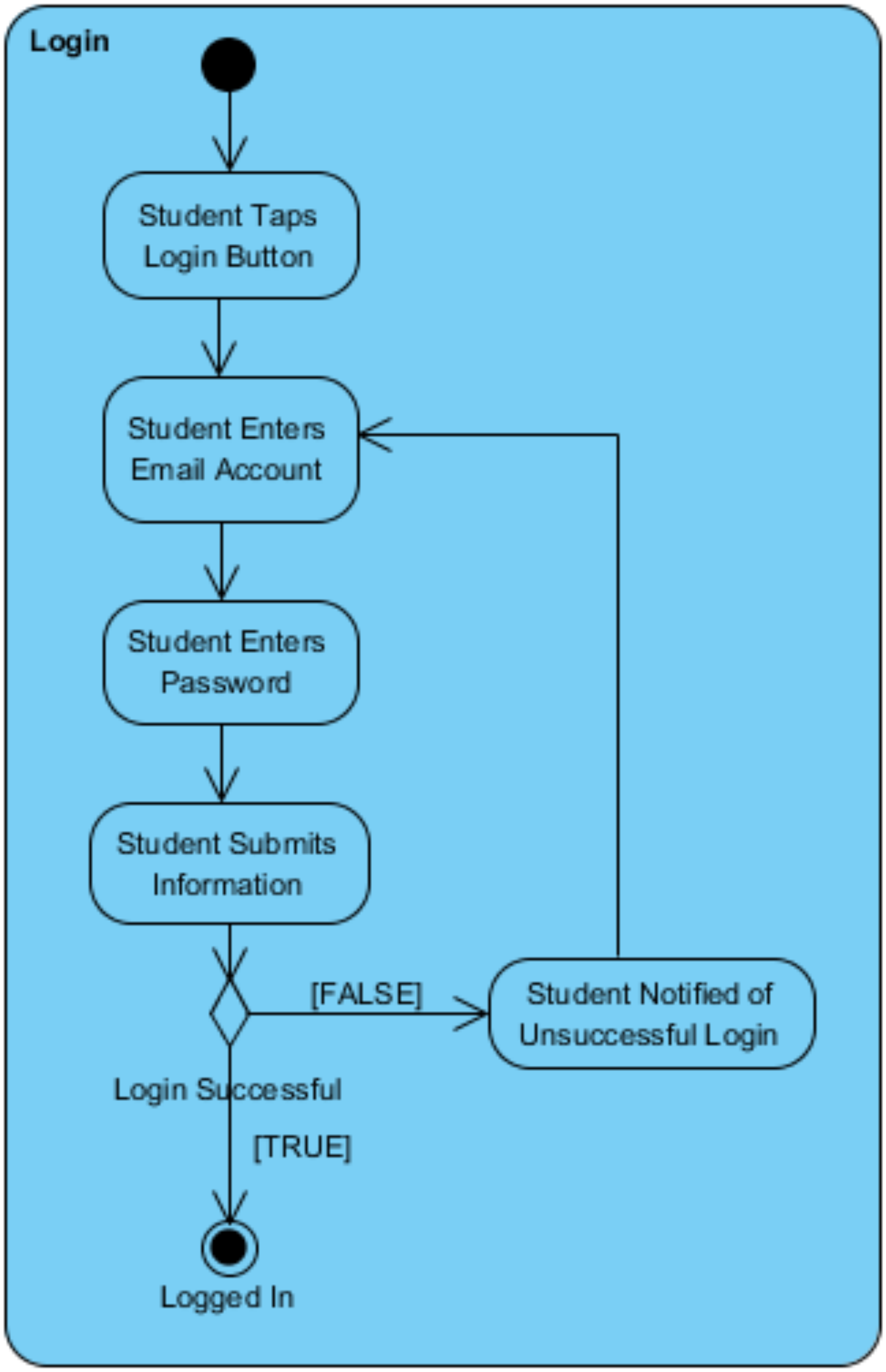
6.4 Rate Teacher

From the users home screen you will have a button to rate a teacher. After tapping the button you will be taken to a search screen where you will enter the professors name. If that professor exists then you will click the name and leave a review for that teacher. If the professor does not exist in the database then you have the option to enter the name into our database and then leave a review for that professor.


Rate Teacher
Initial
Student Taps Rate Teacher Button
Student Taps on Search Teacher Button
Student Enters Teacher's Name
Database Searches for Teacher
[FALSE]
Student Selects Add Teacher to Database
Student Exists in Database
[TRUE]
Student Selects Teacher
Student Writes Teacher Review and Rating
Student Taps Finish Button
[FALSE]
Student Done Rating Teachers
[TRUE]
Teacher(s) Have Been Rated

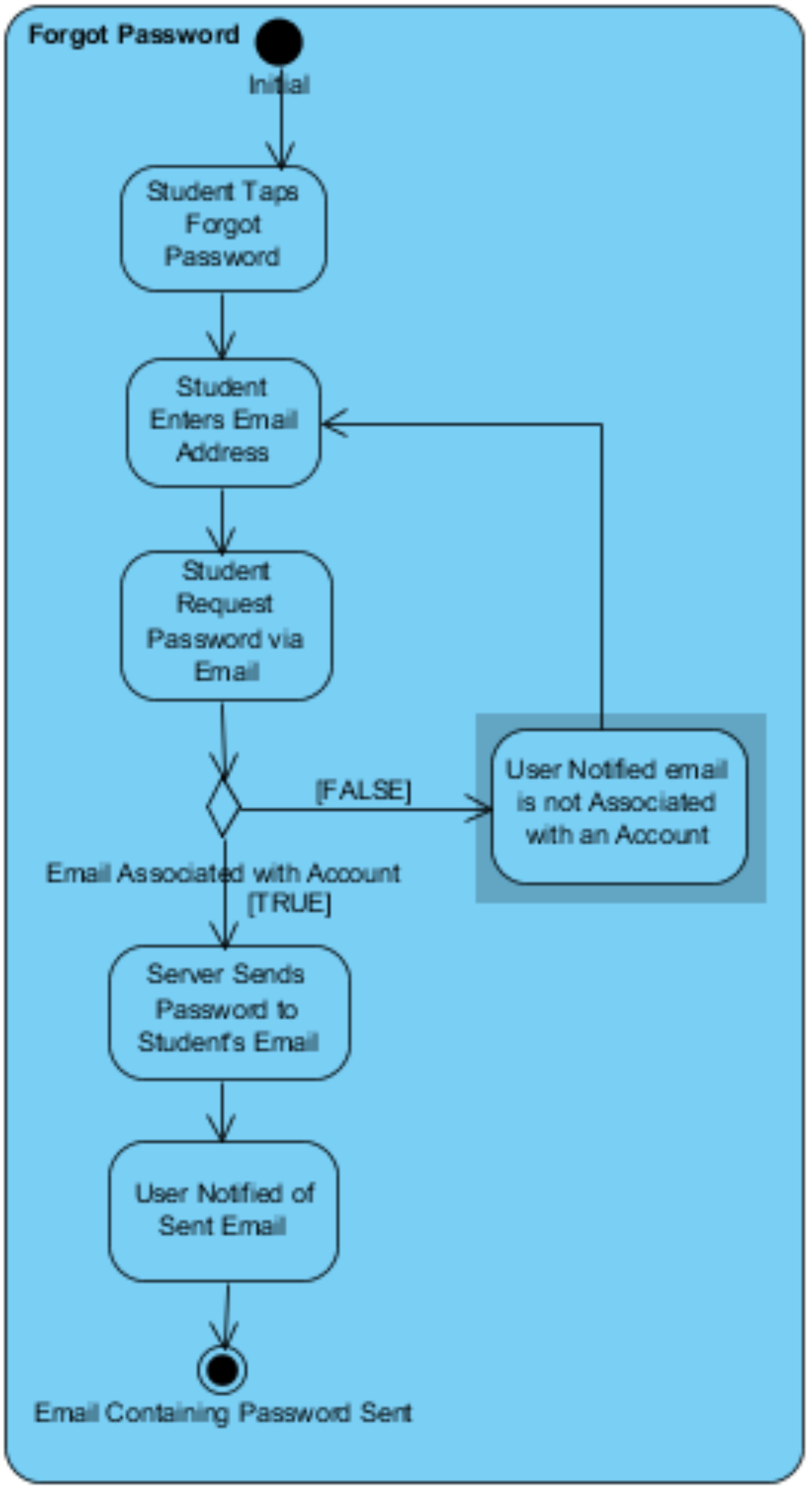
6.5 Login

The main screen will have an to login in and the user will enter their email account and then their password the information is then submitted and if the login was successful they will be taken to their user page. If the login wasn't successful they will be notified to re enter information.


Login
Student Taps Login Button
Student Enters Email Account
Student Enters Password
Student Submits Information
[FALSE]
Student Notified of Unsuccessful Login
Login Successful
[TRUE]
Logged In

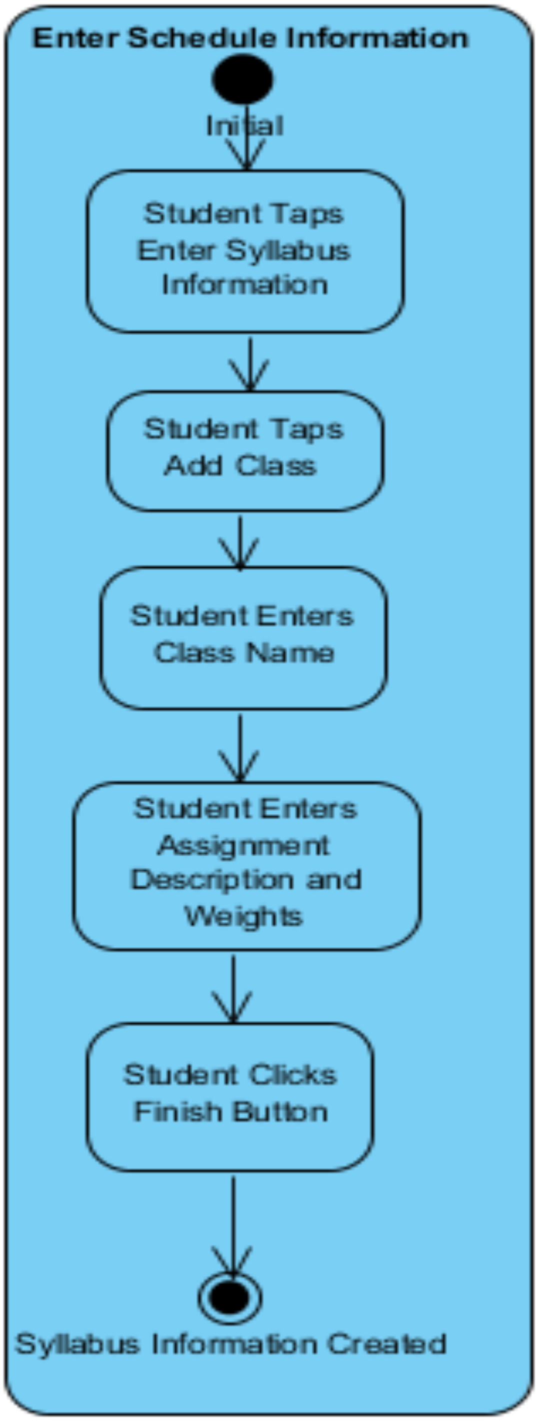
6.6 Forgot Password

On the login screen if the user has forgotten their password they have the option to tap the Forgot Password button and then enter their email address. If the email is associated with an active account then they will receive an email with a temporary password. If the email is not associated with an active account the student has the option to create an account.


Forgot Password
Initial
Student Taps Forgot Password
Student Enters Email Address
Student Request Password via Email
[FALSE]
User Notified email is not Associated with an Account
Email Associated with Account
[TRUE]
Server Sends Password to Student's Email
User Notified of Sent Email
Email Containing Password Sent

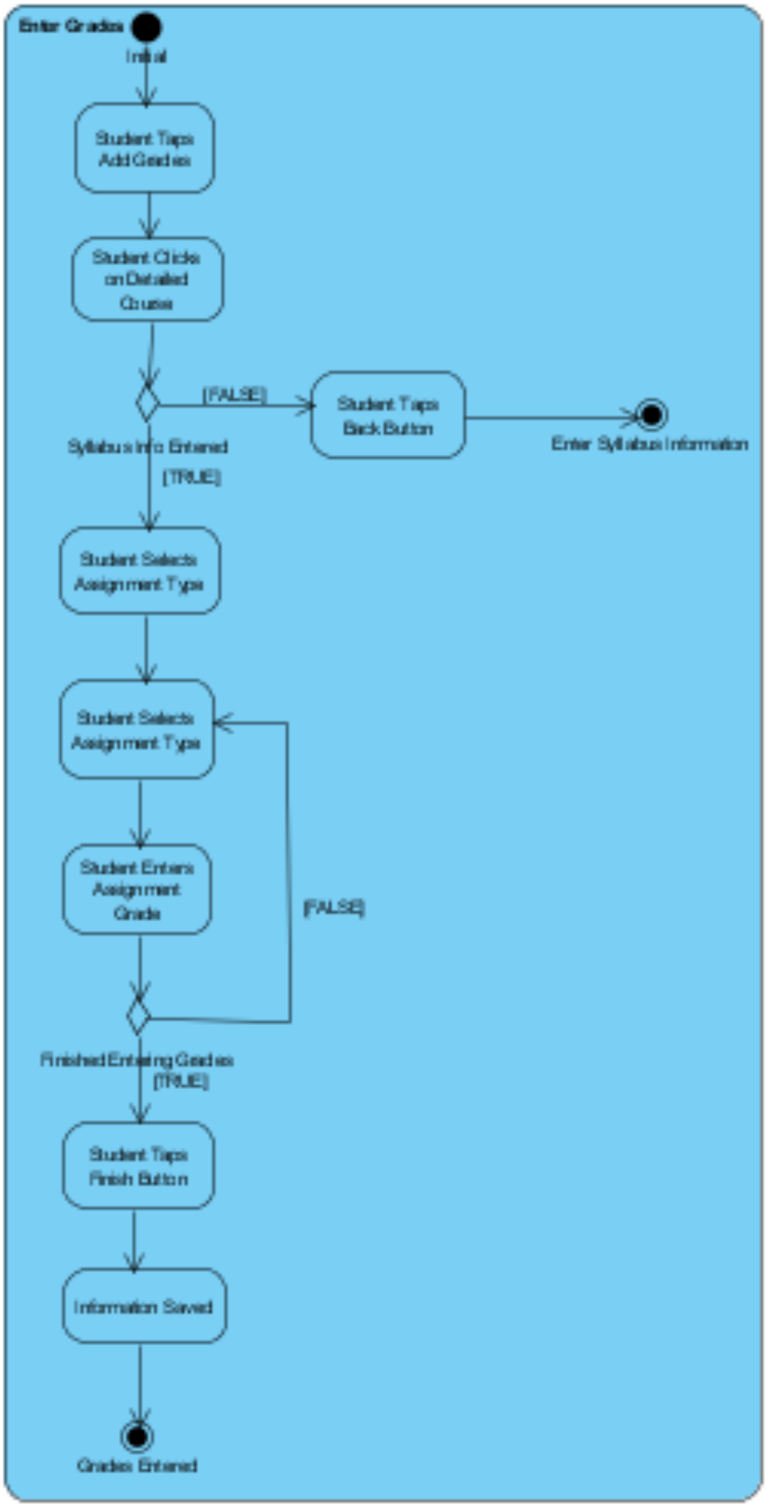
6.7 Enter Schedule Information

If the syllabus information does not already exist the student can choose the class they wish to enter the syllabus for and enter the information manual and then this information will become part of the database for student to use later.


Enter Schedule Information
Initial
Student Taps Enter Syllabus Information
Student Taps Add Class
Student Enters Class Name
Student Enters Assignment Description and Weights
Student Clicks Finish Button
Syllabus Information Created

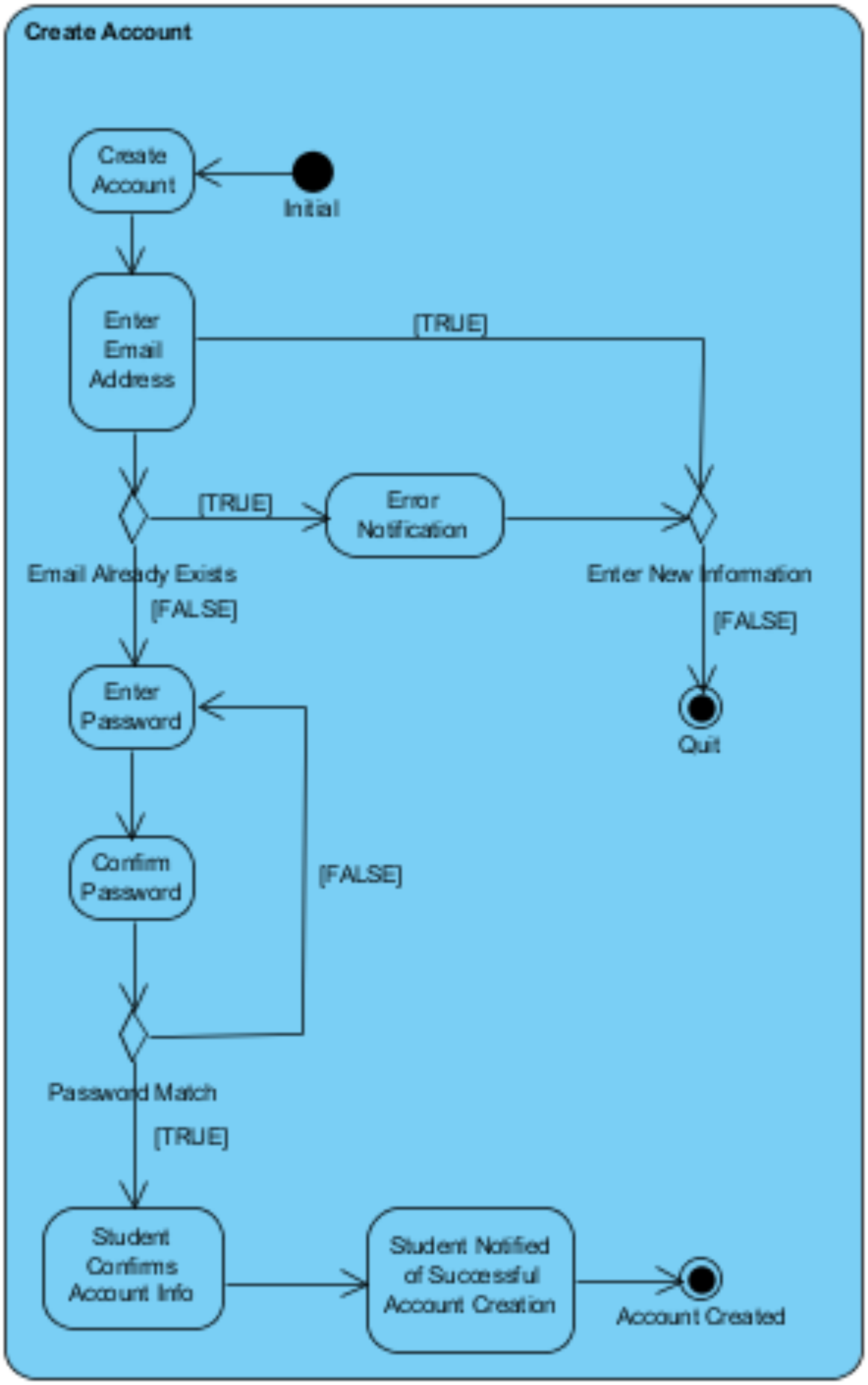
6.8 Enter Grades

From the main page the student can click the Add Grades button and then choose the course they wish to add grades for. Then it will take then t another page with the list of assignments and they can enter the grade for completed assignments.

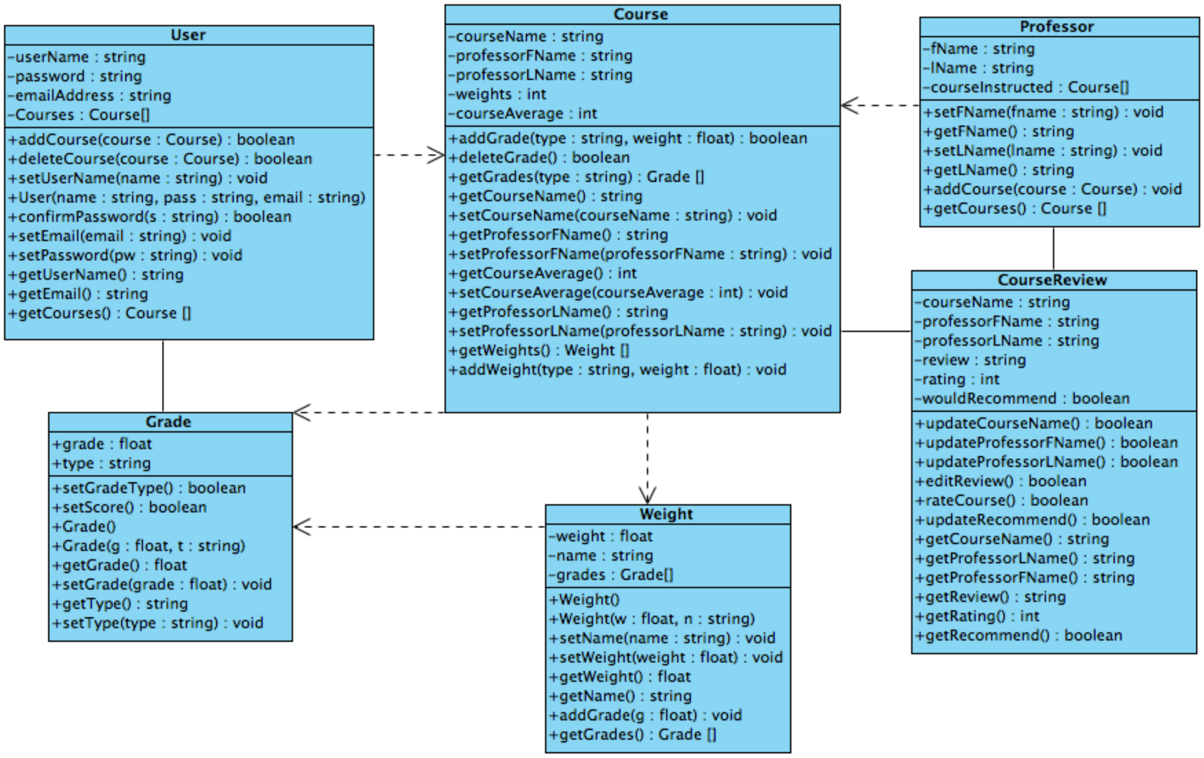

Enter Grades
Initial
Student Taps Add Grades
Student Clicks on Detailed Course
[FALSE]
Student Taps Back Button
Enter Syllabus Information
Syllabus Info Entered
[TRUE]
Student Selects Assignment Type
Student Selects Assignment Type
Student Enters Assignment Grade
[FALSE]
Finished Entering Grades
[TRUE]
Student Taps Finish Button
Information Saved
Grades Entered

6.9 Create Account

If the user is new to the app they have the option to create an account. They enter their email address, which will be checked in our database if it already exists and if it does they have to enter new information. If the email does not exist they enter a password and then reenter password. The user then enter account information and them confirms and they will be sent an email with an account confirmation



7. DETAILED CLASS DIAGRAM

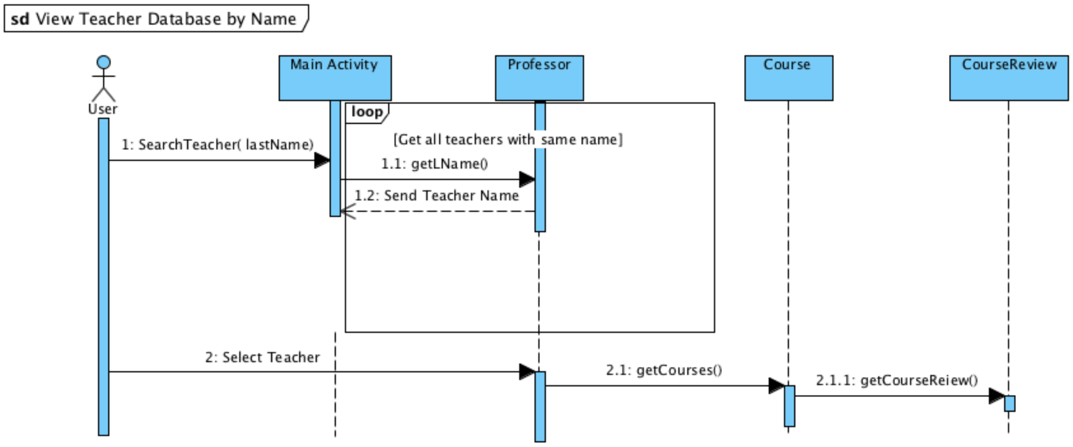

User
-userName : string
-password : string
-emailAddress : string
-Courses : Course[]
+addCourse(course : Course) : boolean
+deleteCourse(course : Course) : boolean
+setUserName(name : string) : void
+User(name : string, pass : string, email : string)
+confirmPassword(s : string) : boolean
+setEmail(email : string) : void
+setPassword(pw : string) : void
+getUserName() : string
+getEmail() : string
+getCourses() : Course []
Course
-courseName : string
-professorFName : string
-professorLName : string
-weights : int
-courseAverage : int
+addGrade(type : string, weight : float) : boolean
+deleteGrade() : boolean
+getGrades(type : string) : Grade []
+getCourseName() : string
+setCourseName(courseName : string) : void
+getProfessorFName() : string
+setProfessorFName(professorFName : string) : void
+getCourseAverage() : int
+setCourseAverage(courseAverage : int) : void
+getProfessorLName() : string
+setProfessorLName(professorLName : string) : void
+getWeights() : Weight []
+addWeight(type : string, weight : float) : void
Professor
-fName : string
-lName : string
-courseInstructed : Course[]
+setFName(fname : string) : void
+getFName() : string
+setLName(lname : string) : void
+getLName() : string
+addCourse(course : Course) : void
+getCourses() : Course []
CourseReview
-courseName : string
-professorFName : string
-professorLName : string
-review : string
-rating : int
-wouldRecommend : boolean
+updateCourseName() : boolean
+updateProfessorFName() : boolean
+updateProfessorLName() : boolean
+editReview() : boolean
+rateCourse() : boolean
+updateRecommend() : boolean
+getCourseName() : string
+getProfessorLName() : string
+getProfessorFName() : string
+getReview() : string
+getRating() : int
+getRecommend() : boolean
Grade
+grade : float
+type : string
+setGradeType() : boolean
+setScore() : boolean
+Grade()
+Grade(g : float, t : string)
+getGrade() : float
+setGrade(grade : float) : void
+getType() : string
+setType(type : string) : void
Weight
-weight : float
-name : string
-grades : Grade[]
+Weight()
+Weight(w : float, n : string)
+setName(name : string) : void
+setWeight(weight : float) : void
+getWeight() : float
+getName() : string
+addGrade(g : float) : void
+getGrades() : Grade []

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.

8. SEQUENCE DIAGRAM

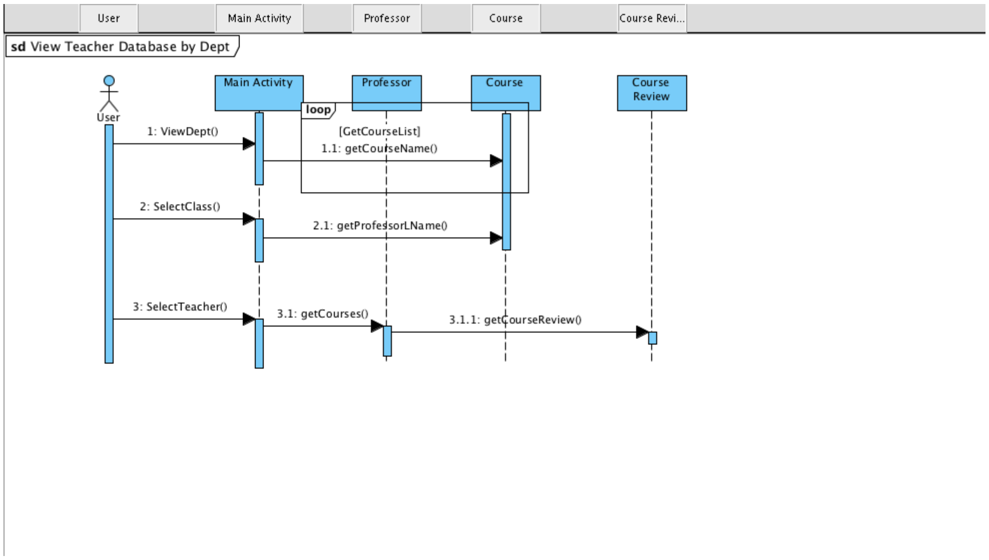
8.1 View Teacher Database by Name

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.



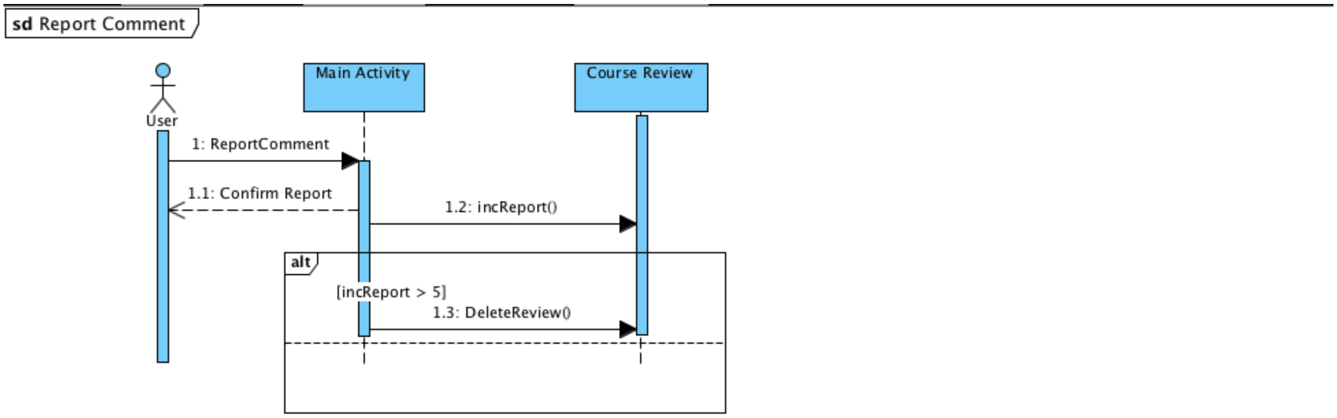
8.2 View Teacher Database by Dept

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



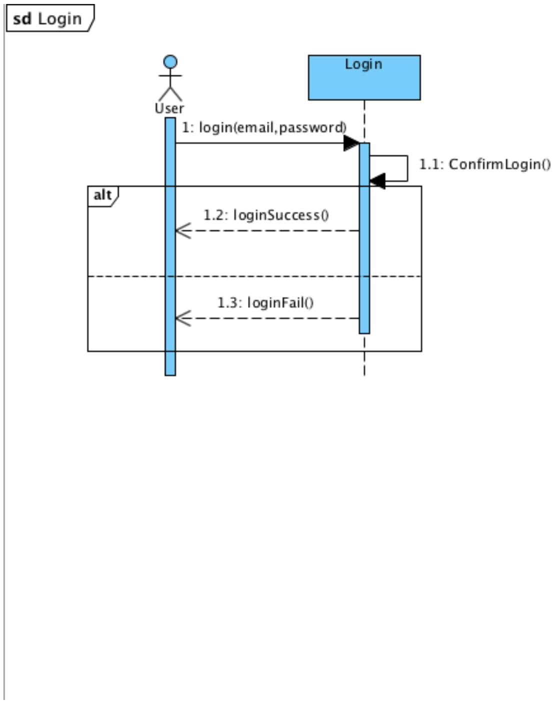
8.3 Report Comment

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.



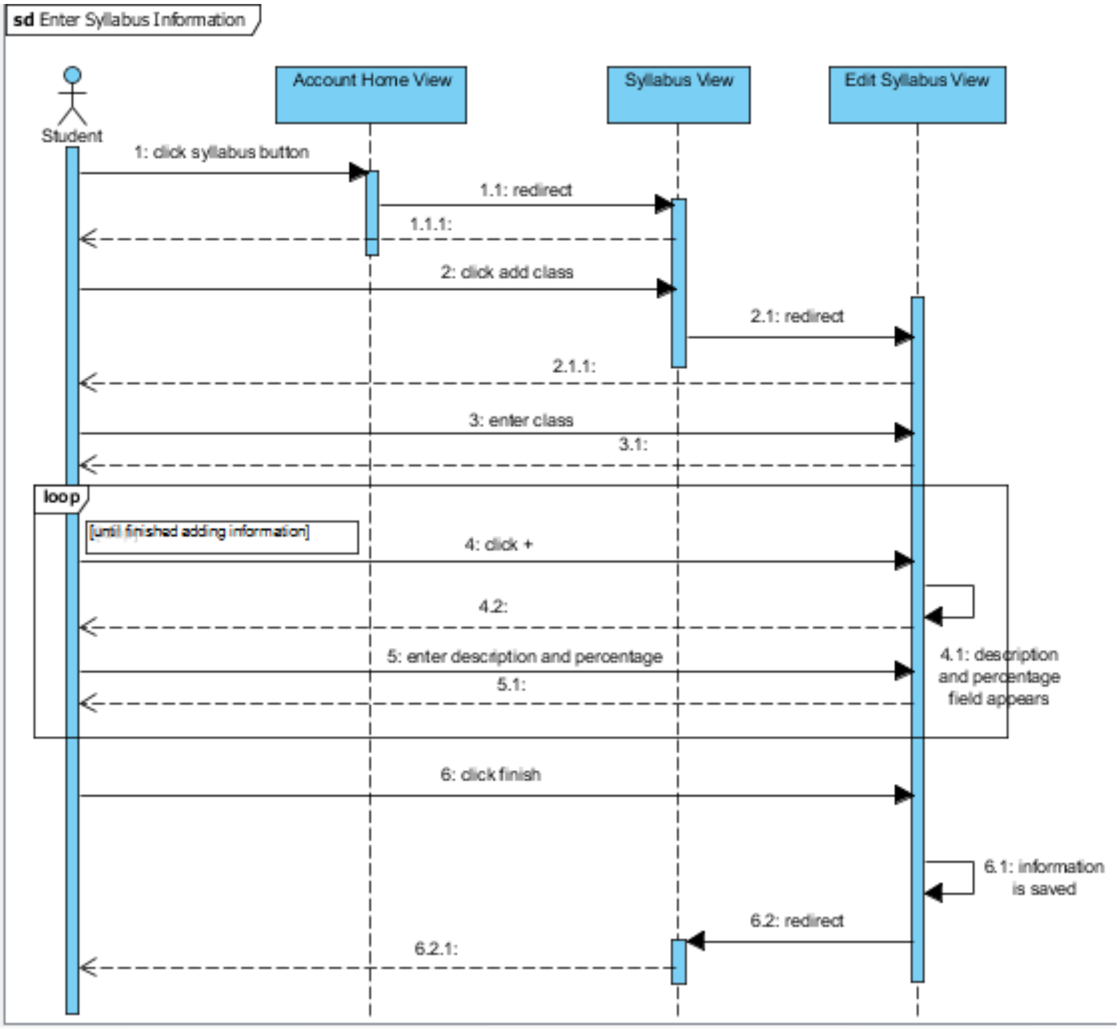
8.4 Login

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



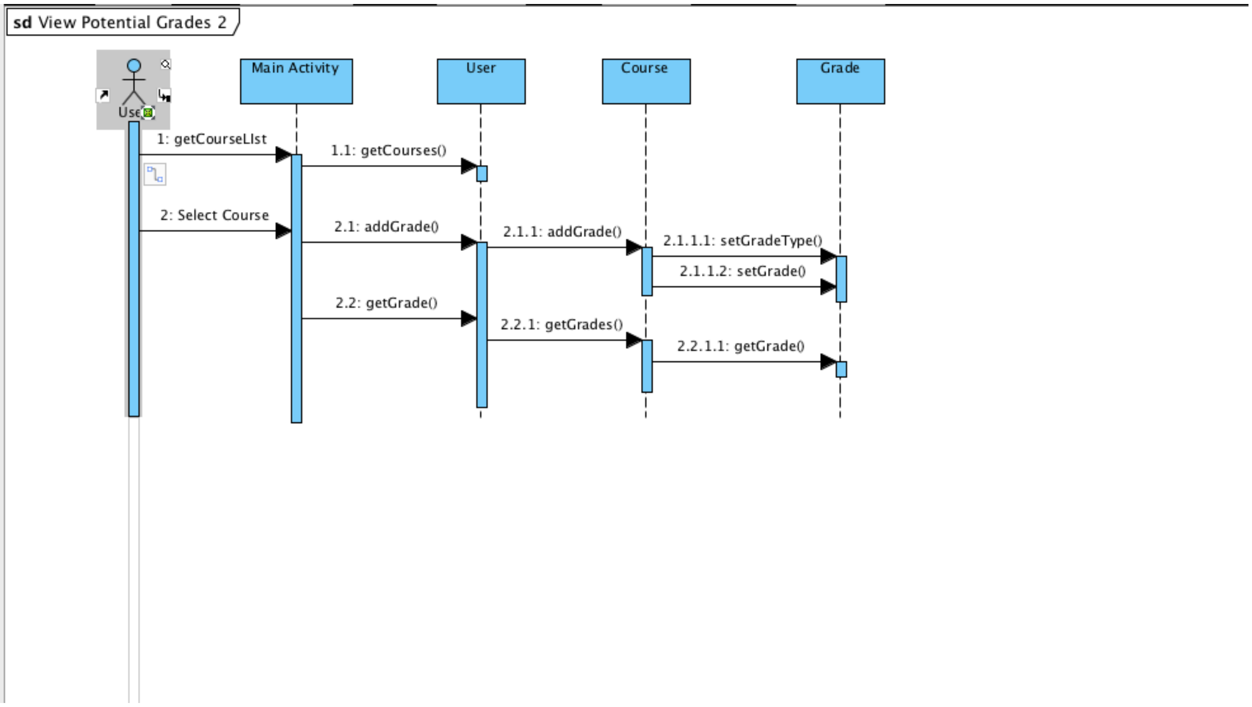
8.5 Enter Syllabus Information

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.



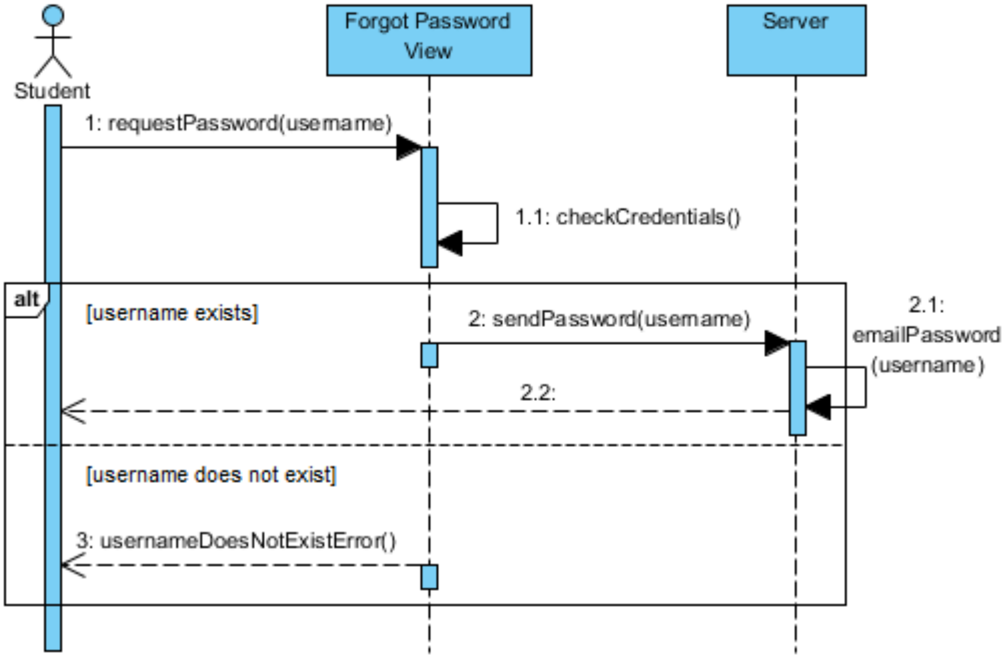
8.6 View Potential Grades 2

This sequence diagram illustrates the process for checking what-if scenarios to determine what the user's grade can potentially be.



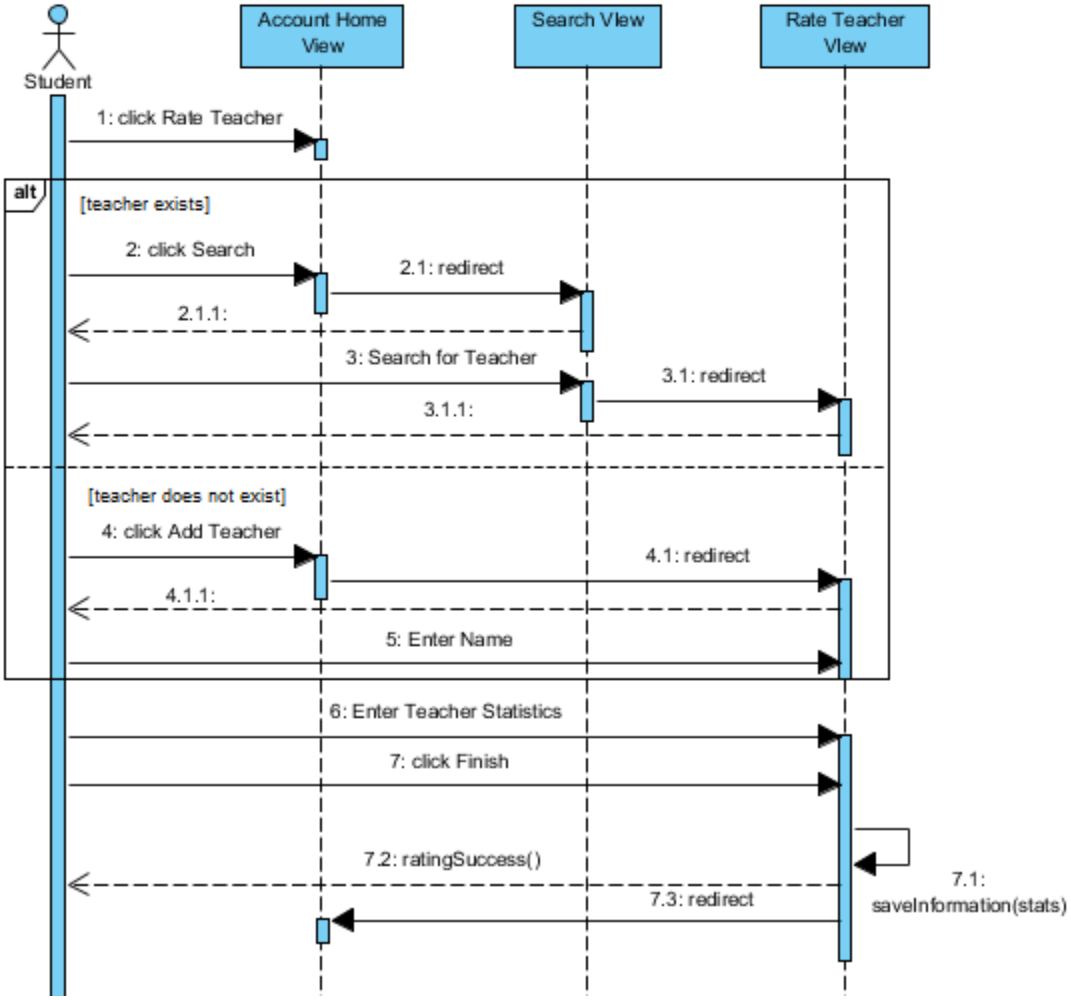
8.7 Forgot Password

This diagram illustrates the use case where a user will request to have their password emailed to them. If the account specified exists it will be emailed to them.


Student
Forgot Password View
Server
1: requestPassword(username)
1.1: checkCredentials()
alt
[username exists]
2: sendPassword(username)
2.1: emailPassword (username)
2.2:
[username does not exist]
3: usernameDoesNotExistError()

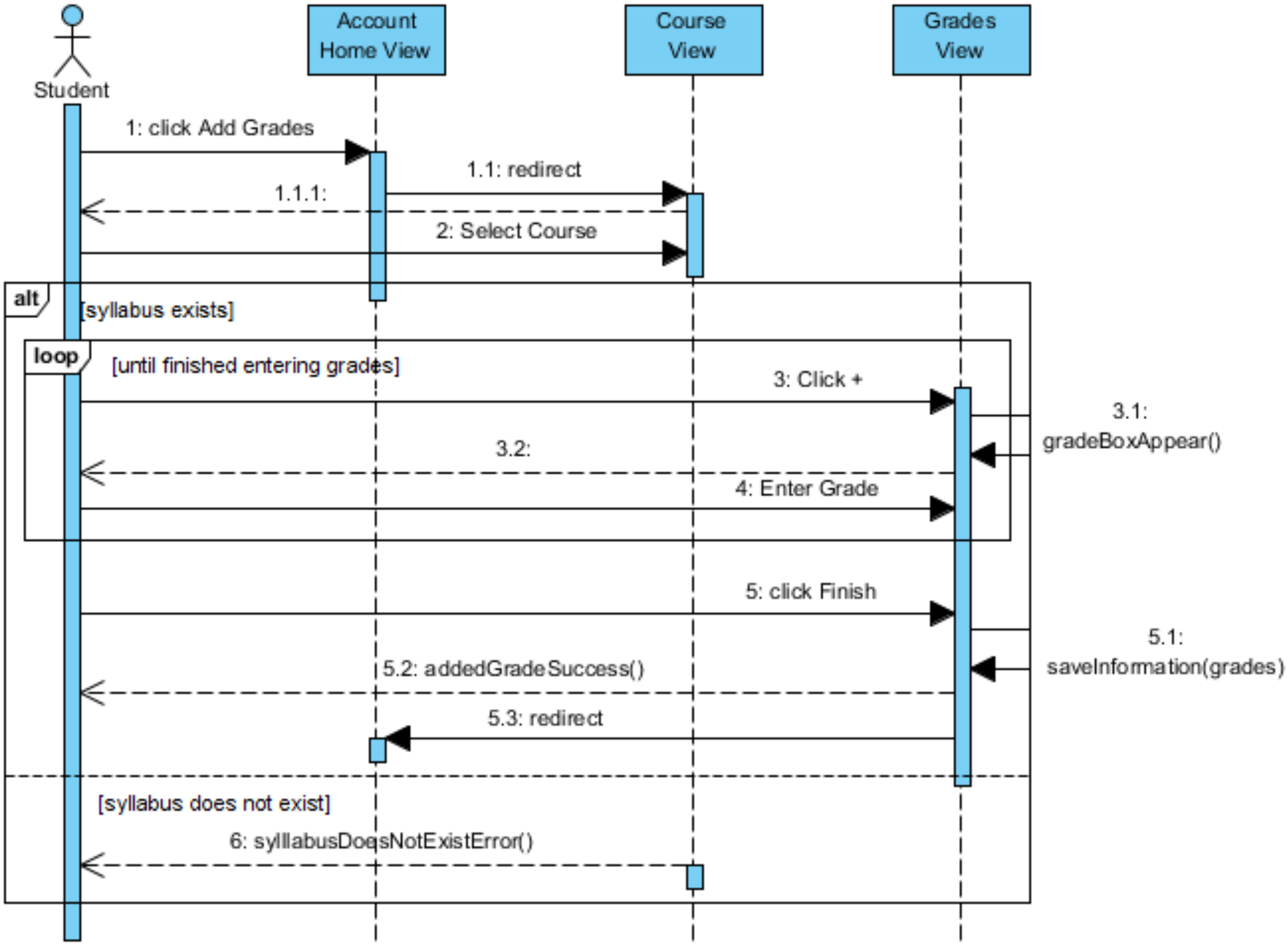
8.8 Rate Teacher

This diagram illustrates the use case where a student rates their professor. The student is able to search for the professor they wish to rate and enter their rating of them.



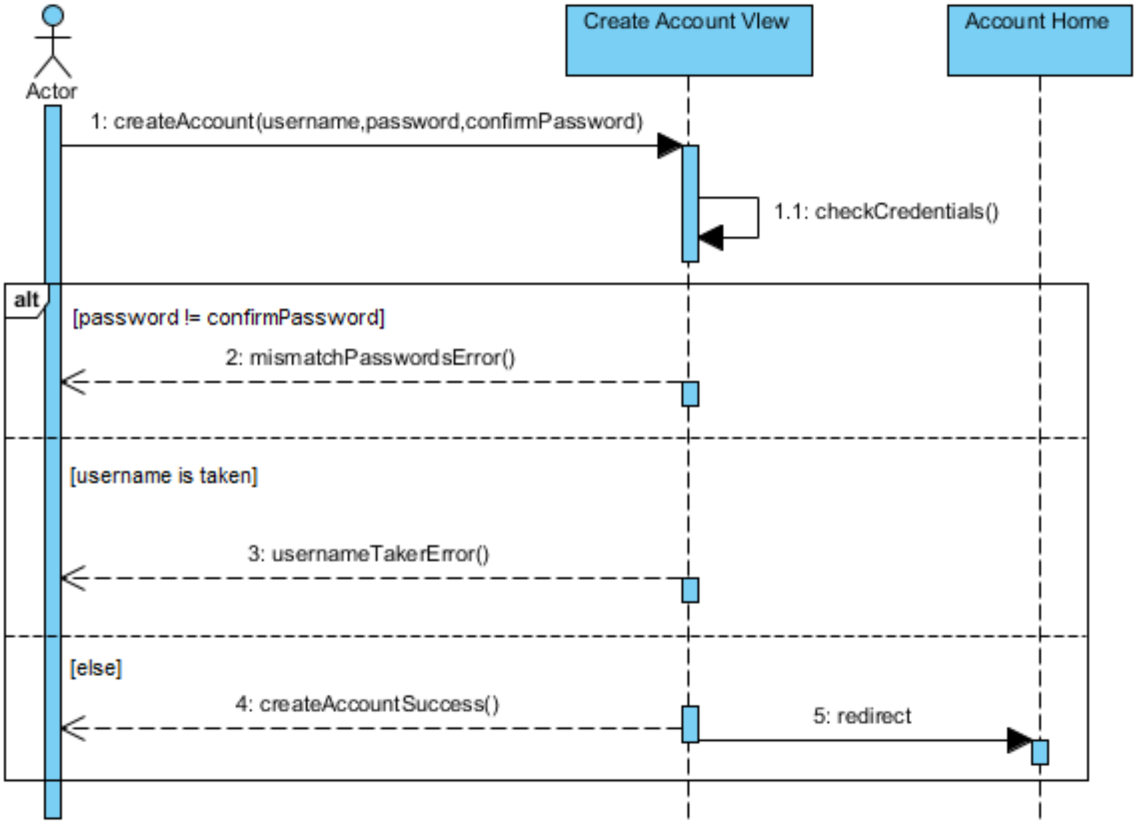
8.9 Add grades

This diagram illustrates the use case where a student enters grade for a course. They will initially need to create a syllabus, otherwise they will not be able to enter their grades.



8.10 Create Account

This diagram illustrates the use case where a student creates their account. The process will make sure no duplicate accounts are created and that the passwords entered match up.


Actor
Create Account VIew
Account Home
1: createAccount(username,password,confirmPassword)
1.1: checkCredentials()
alt
[password != confirmPassword]
2: mismatchPasswordsError()
[username is taken]
3: usernameTakerError()
[else]
4: createAccountSuccess()
5: redirect