FailSafe

Software Requirements Document

### Autumn Jackson

### Bre Tucker

### Jordan Gill

### 9/22/2016

Change History:

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Summary | Author | Date |
| 0.1 | Initial Write-up | Jordan Gill | 9/21/2016 |
| 0.2 | Added to functional and non-functional requirements | Jordan Gill | 9/21/2016 |
| 0.3 | Added Class diagram and descriptions | Autumn Jackson | 9/22/2016 |
| 0.4 | Added Activity diagrams and descriptions | Autumn Jackson | 9/22/2016 |
| 0.5 | Added Use case diagrams and descriptions | Bre Tucker | 9/22/2016 |
| 0.6 | Reformatting | Jordan Gill | 9/22/2016 |
| 0.7 | Added Sequence Diagrams | Jordan Gill | 10/20/2016 |
| 0.8 | Added Project Description | Jordan Gill | 10/20/2016 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[Autumn Jackson 1](#__RefHeading___Toc543_1660731095)

[Bre Tucker 1](#__RefHeading___Toc545_1660731095)

[Jordan Gill 1](#__RefHeading___Toc547_1660731095)

[9/22/2016 1](#__RefHeading___Toc549_1660731095)

[1. Introduction 1](#__RefHeading___Toc303_88834589)

[1.1 Motivation/purpose 1](#__RefHeading___Toc305_88834589)

[1.2 Scope 1](#__RefHeading___Toc307_88834589)

[1.3 Goals 1](#__RefHeading___Toc309_88834589)

[1.4 Definitions 2](#__RefHeading___Toc510_397140797)

[2. Project Description 2](#__RefHeading___Toc311_88834589)

[2.1 Add Classes and Assignments 2](#__RefHeading___Toc321_88834589)

[2.3 What-If 3](#__RefHeading___Toc323_88834589)

[2.5 Map 3](#__RefHeading___Toc325_88834589)

[2.6 Multiuser Support 4](#__RefHeading___Toc327_88834589)

[3. System Requirements 4](#__RefHeading___Toc313_88834589)

[3.1 Functional Requirements 4](#__RefHeading___Toc329_88834589)

[3.2 Non-Functional Requirements 5](#__RefHeading___Toc331_88834589)

[5. Diagrams 6](#__RefHeading___Toc315_88834589)

[5.1 Use Case Diagrams 6](#__RefHeading___Toc333_88834589)

[5.2 Class Diagram 8](#__RefHeading___Toc335_88834589)

[5.3 Activity Diagrams 8](#__RefHeading___Toc337_88834589)

[5.4 Sequence Diagrams 13](#__RefHeading___Toc495_397140797)

# 1. Introduction

## 1.1 Motivation/purpose

The purpose of the application FailSafe is to give students the ability to more actively track their grades and overall GPA and thus stay on top of their class performance. The motivation behind developing FailSafe for mobile devices is to increase accessibility to students, who are often on the go and not always near a computer, which in turn will ideally encourage use of the application to track grades and assignments.

## 1.2 Scope

By the end of development, FailSafe will be able to accurately calculate the grades, weighted and unweighted, of any and all assignments, quizzes, and exams. The application will also allow users to drop grades and recalculate their overall grade for a course, as well as feature a “What If” option that will give users the opportunity to calculate potential grades with hypothetical assignment scores. As a matter of course, FailSafe will also be able to keep track of multiple classes and the coursework and grades associated with them.

## 1.3 Goals

The goals for the first version of the project are to implement the class and coursework tracking features and the grade calculator. The goal of the second version of the application is to include a functional in-app calendar to which assignments can be automatically added. A calendar native to the application will allow for more information about each assignment to be displayed, as well as allowing for filtering options such as displaying work for only once class on the calendar as opposed to all classes. Finally, the goals for a third version include multiuser support, professor-specific features such as allowing professors to submit class-wide assignments or grade weights, and an interactive map to assist users in finding specific buildings.

## 1.4 Definitions

* Assignment: Work assigned by instructor; includes homework, tests, labs, in class assignments, projects, and quizzes.
* Drops: Grades that can be excluded from the calculated grade.
* Weight: The percentage that each assignment is worth.
* Gradebook: Class within FailSafe that calculates and holds all grades.

# 2. Project Description

* 1. The main functionality of the system will include allowing users to add classes and assignments including any grade weights associated with the assignments, and save and calculate overall class grades as well as calculate averages for specific assignments sets. In addition, a “what-if” feature, teacher specific tools, a campus map, and multi-user support will also make up a portion of the application.

## 2.1 Add Classes and Assignments

First and foremost, FailSafe will allow users to input their class schedules and any coursework associated with each class. Users will also be able to add grade weights for each course and due dates for all assignments. In addition to basic assignments such as homework and readings, students will able to enter and track exams and quizzes, including the related dates.

2.2 Save and Calculate Grades

The application will save any inputted grades, unless removed by the user, to be searched and viewed at any time. These saved grades are then used to calculate a running overall grade for the class in question. If grades have been entered for the course, then the grade will automatically be calculated with the submitted weights. Otherwise, the calculated score will be a basic unweighted average of the saved scores.

## 2.3 What-If

This feature will allow students to calculate potential grades based on hypothetical scores. The feature will extend to both individual classes as well as overall semester GPA calculation based on the courses saved in the application.

2.4 Teacher Tools

Though FailSafe is primarily intended to be used by students, various tools can be added to facilitate use for educators. These features will include the ability for the professor to add grades and assignments to the class as a whole, so that students in the class will not have to do so individually.

## 2.5 Map

To aid students in finding the classes they have inputted, FailSafe will also provide an interactive campus map. Using GPS, the application will track the user with a marker on the map and lead them to the desired building. If no GPS is available, then a highlighted path from a position of the user’s choosing to the end point will be shown.

## 2.6 Multiuser Support

Ultimately, FailSafe will be able to support multiple users and allow them to interact with one other in various ways. User will theoretically be able to chat with other users, share notes and study materials, and interact with professors using the app. To this end, a username and password system would be implemented.

# 3. System Requirements

## 3.1 Functional Requirements

The application will be expected to do the following:

1. Allow users to add, remove, and keep track of individual courses.
2. Allow users to add, remove, and keep track of all coursework, including homework, readings, exams, and quizzes, associated with each class.
3. Keep track of due dates for each assignment.
4. Allow users to add grade weights for each course.
5. Allow users to add, save, and calculate weighted and unweighted grades for each course.
6. Allow users to calculate overall GPA using the submitted classes.
7. Allow users to calculate potential grades using the “What-If” feature.
8. Automatically add coursework to calendar and display relevant information (to be implemented if time permits).
9. Display a campus map to assist users with finding buildings (to be implemented if time permits)

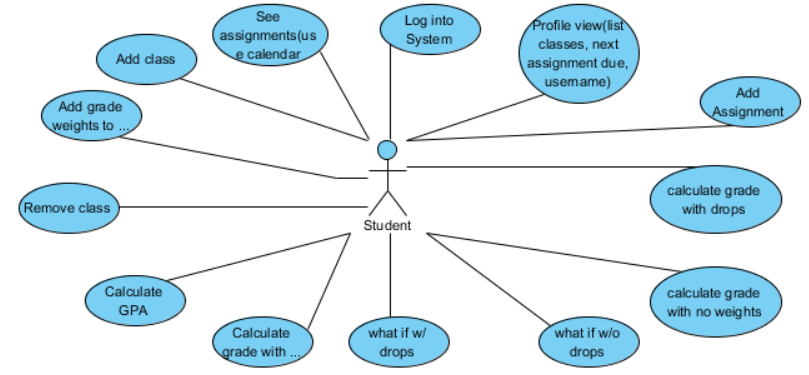
## 3.2 Non-Functional Requirements

The application will be expected to perform in the following manner:

* Grade calculation should be performed quickly and accurately
* The application should work smoothly on all Android devices (other mobile device support to be added at a later time).
* Username and password combinations will be used to ensure basic security and privacy for all users.
* The application should be user friendly and intuitive to use. A user should be able to use the application to its full potential immediately upon downloading.
* The application should be easily modifiable to expand to other universities.
* The application will be largely self-contained and should not draw heavily on the target device’s resources.
* There can be no unhandled exceptions from incorrect user input.
* All menus must have a consistent format.
* During a system restart, the system will return to a functioning state.

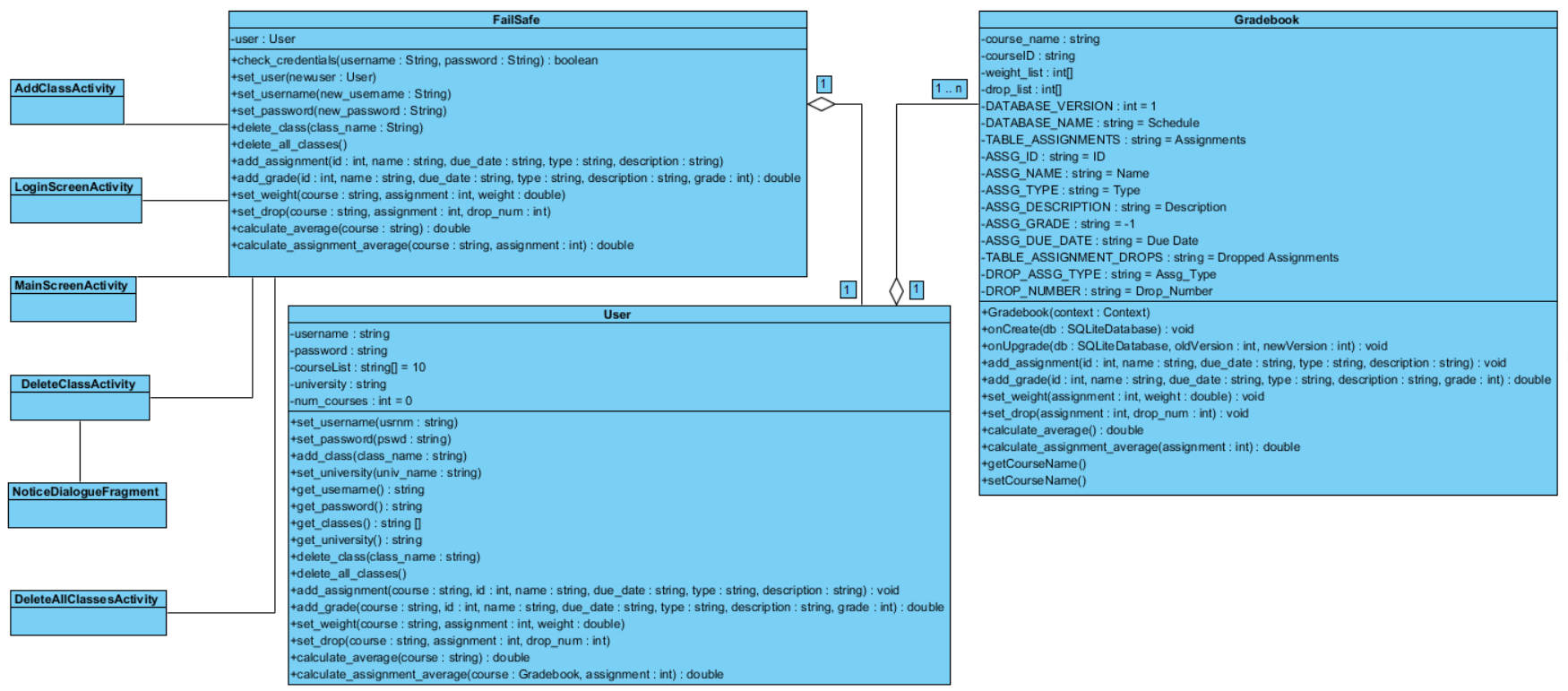
# 5. Diagrams

## 5.1 Use Case Diagrams



Our main Actor in FailSafe is the student, in the future we would like to add a professor as an additional user. The user is able to log into the system to get to the profile view. From this view he/ she is able to view the list of classes, their next assignment, their due date, and username. The student is also able to see assignments via the calendar. From the profile view, a student is able to add a course to their profile. When a course is added the student is then able to add assignments for that course, and also add grade weights to their assignments. The student is also given the option to remove a class if he/she would like. Once a student adds a course and adds the assignments and weights he/she is given the options to see their calculated grades. The user is also given the option to see a calculated grade with the option to choose a calculated grade with no drops and one with drops. They are also given the opportunity to generate a calculated grade with a what-if function where the user can calculate a grade with chosen scores to see their progress in the course before all assignments are graded. Finally, the user is given a option to see their calculated GPA once the scores have been all inputted.

## 5.2 Class Diagram



Our system three main classes and miscellaneous classes that allow the user to interact with those main classes. At the moment we have five miscellaneous classes: AddClassActivity, LoginScreenActivity, MainScreenActivity, DeleteClassActivity, and NoticeDialogueFragment. All of them represent interactive screens in the app and all except NoticeDialogueFragment enable the user to interact with FailSafe directly by passing a serialized version of the current FailSafe object between them as necessary, but more on that later.

LoginScreenActivity controls the buttons for… the login screen! It reacts to the user clicking “Create User” to create a new account and also facilitates checking their login credentials. LoginScreenActivity launches NewUserActivity and MainScreenActivity.

MainScreenActivity acts as the main hub of the app, the Grand Central Station. It contains the interactive methods for adding a class, deleting a class, and deleting all classes. It also contains the serialized FailSafe object, which is passed to it from LoginScreen. Eventually, it will hopefully have on screen buttons for each class. This class launches AddClassActivity, DeleteClassActivity, and DeleteAllClassesActivity.

AddClassActivity is passed the FailSafe object and allows a user to add classes to FailSafe.

DeleteClassActivity is passed the FailSafe object and does the opposite of the previous class. It allows the user to delete classes from FailSafe. It confirms whether the user really meant to delete this class by launching the NoticeDialogueFragment class which handles the pop-up dialogue box.

DeleteAllClassesActivity is basically the same as DeleteClassActivity but it allows the user to delete all their classes (imagine that).

The FailSafe class contains methods for allowing the user to interact with the User and their Gradebooks. Accordingly, it contains all the public methods for Gradebook and those for User as well and those methods will be discussed with their respective classes. FailSafe’s only unique method is check\_credentials, where it checks whether the username and password combination entered by the user matches the username and password stored in its User variable.

For attributes, the User class has a username, password, university (for use later when FailSafe can contain more than one User, after we get all the necessaries done), courseList, and num\_courses. Username, password, and university are all strings and pretty self-explanatory. CourseList is an array that can hold 10 Gradebook objects. We chose 10 because most users shouldn’t have more than 7 or 8 classes. Num\_courses acts as a pointer to where to store the next new Gradebook and reflects how many classes the user has stored.

As for User methods (other than getters and setters):

* add\_class(String class\_name): makes a new Gradebook object with name = class\_name, stores it in courseList[num\_courses] and increments num\_courses.
* delete\_class(String class\_name): loops thru courseList until it finds the Gradebook with the matching name and makes the necessary changes to remove it from the system while still keeping courseList contiguous.
* delete\_all\_classes( ): removes all classes from the system
* As for the calling methods for Gradebook, the only way they differ from the one’s in grade book is that they each have the extra parameter “course”. It’s a string that allows User to search the courseList until it find the matching Gradebook then call that class’s version of the method.

Now for Gradebook. We got rid of the Course class since the only thing that differed from Gradebook was two additional attributes of course\_name and courseID. Now Gradebook has those 2 attributes. It’s REALLY a work in progress, but the attributes and methods are as follows:

Attributes:

* course\_name: the name of the class
* courseID: the class’s course ID
* weight\_list: an array of size 6 that holds the associated weights for tests, homework, lab, in class assignments, projects, and quizzes, in that order.
* DATABASE\_NAME: the name of the database
* 2 database tables: TABLE\_ASSIGNMENTS and TABLE\_DROPS
* TABLE\_ASSIGNMENTS column names:
  + ASSG\_ID: unique identifier for the assignment
  + ASSG\_NAME: name of assignment
  + ASSG\_TYPE: type of assignment (test, homework, etc.)
  + ASSG\_DESCRIPTION : short description of the assignment
  + ASSG\_GRADE: grade received for the assignmentinitialised to -1 (because while you can have a 0, a professor would have to be pretty evil to give a negative assignment grade)
  + ASSG\_DUE\_DATE: when the assignment is due
* TABLE\_DROPS column names:
  + DROP\_NUMBER: number of drops allowed of this type
  + DROP\_ASSG\_TYPE: type of assignment (test, homework, etc.)

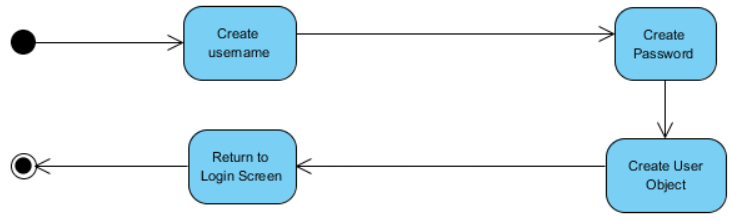
Methods:

* add\_assignment(int id, String name, String due\_date, String type, String description): adds a row to the database with these attributes
* add\_grade(int id, String name, String due\_date, String type, String description, String grade): updates a row to the database with the new grade value
* set\_weight(int assignment, double weight): sets the assignment weights
* set\_drops(int assignment, int drop\_num): sets the number of drops in TABLE\_DROPS with its associated assignment type
* calculate\_average(): calculates the student’s average for this course
* calculate\_assignment\_average(int assignment): calculate the student’s average for a particular assignment set (i.e. JUST their tests, or JUST quizzes)

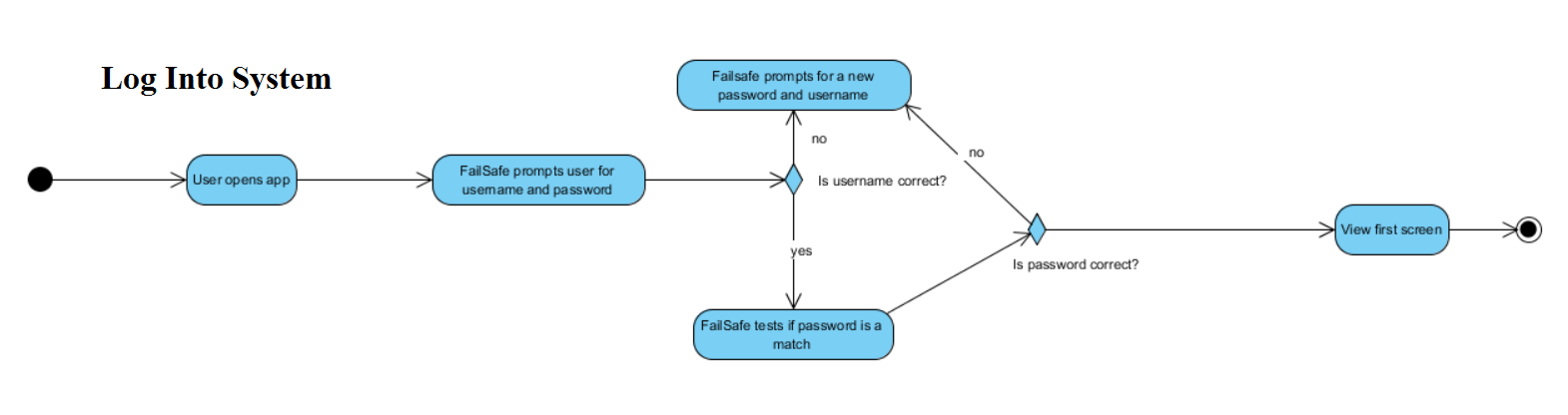
## 5.3 Activity Diagrams

We have five activities within the FailSafe system: creating a user, login, adding a class, adding an assignment, calculating a grade, and removing a class.

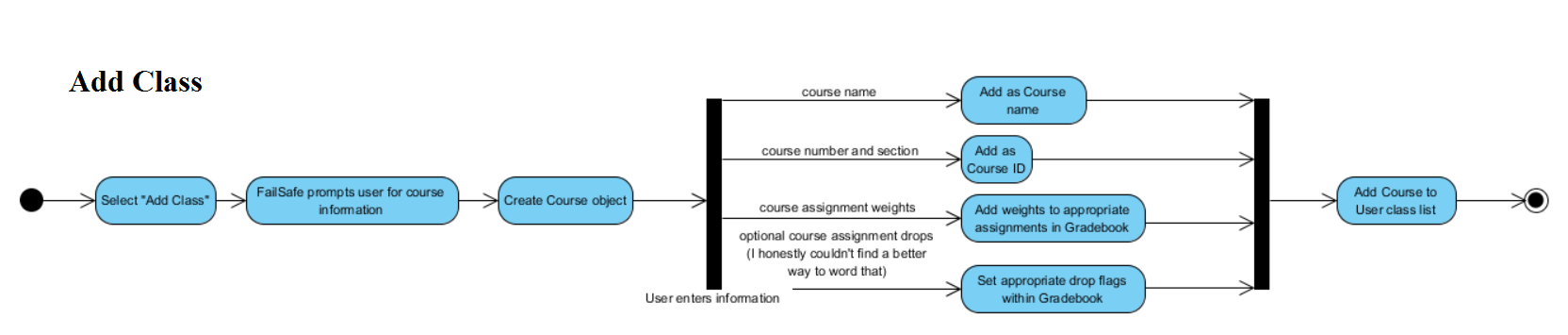
**Create User.** When the user opens the app, they’re presented with a screen with username/password boxes and a “Create User” button. Since at this point the user doesn’t have a username or password, they will choose the “Create User” button. It takes them to a new screen where they will create a username and password. Once they tap the “Add User” button, FailSafe creates a User object where it stores the credentials then returns the User to the login screen.



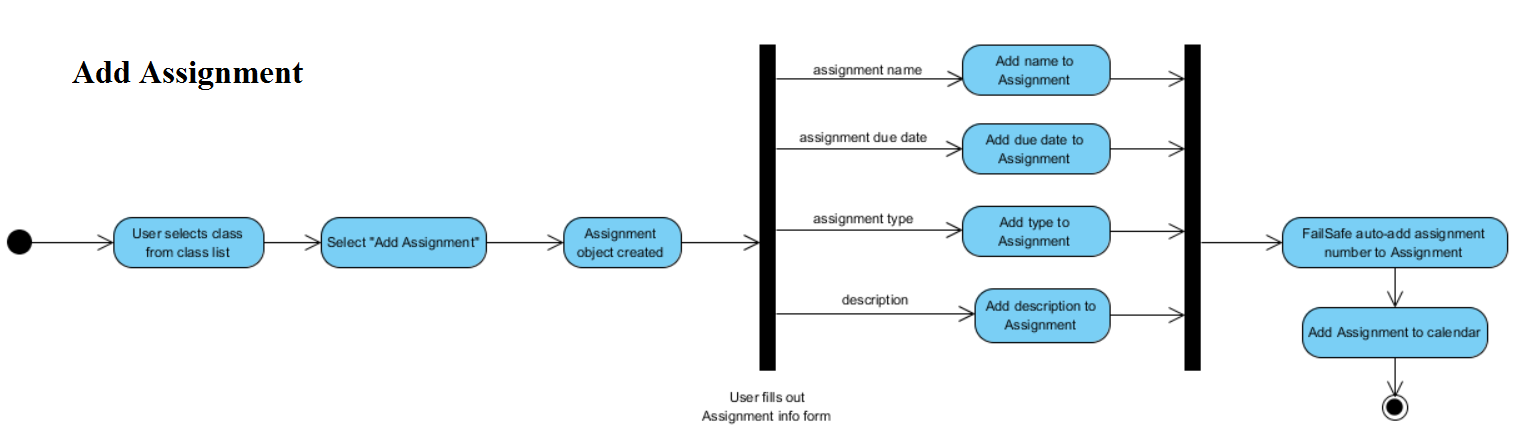
**Log Into System.** After the user creates login credentials, they need to use them. The first action of the Log Into System activity is the user opens the app, but they just created an account in the above activity, so they are already on the open screen where FailSafe prompts them with the username/password boxes. The user enters their username and password and hits enter. Failsafe checks the username first. If it is incorrect, FailSafe prompts them for a new username and password. If it is correct, Failsafe then checks whether the password stored with that username matches the one submitted. If it does not, FailSafe prompts for a new username and password. Otherwise, the user is taken to the first screen within Failsafe.



**Add Class.**Next, the user needs to add a class to FailSafe. They select “Add Class” on the welcome screen and are presented with a form page prompting them for the course information. Behind the scenes, FailSafe creates a Course object to store the information the user will provide. The user enters the course name (stored as *name*), course number & section (stored as *course\_id*), the weights for the various types of assignments in the class (added to their appropriate places in Gradebook), and indicates which assignments, if any, have the option of dropping grades from the average and how many (sets appropriate flags within Gradebook). Once the user has entered all the required information, FailSafe adds the Course object to the user’s class list in their User object.



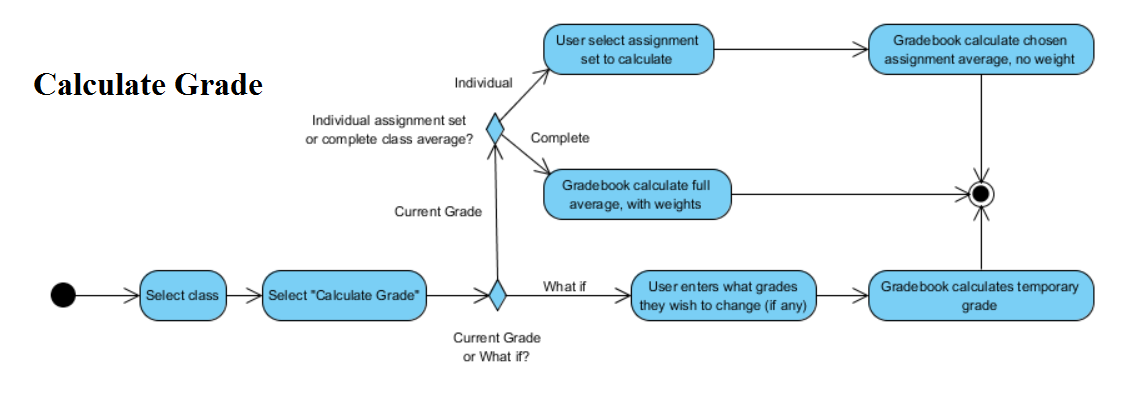
**Add Assignment.** Now that they have added a class, the user needs to enter their assignments or they will not be able to calculate any grades, and that is FailSafe’s main objective. To add an assignment to a class, the user first selects the desired class from their class list and from that class’s screen chooses “Add Assignment”. Similar to the Add Class activity, FailSafe creates an Assignment object and the gives the user a form page to fill out with the assignment information. The user enters the assignment’s name, due date, type (test, reading, project, etc.), and a description of the assignment. All of these entries are then stored in their designated places within the Assignment object. FailSafe assigns the assignment a number based on how many assignments of that type have been added before it, then adds the assignment into the calendar and exits the activity.



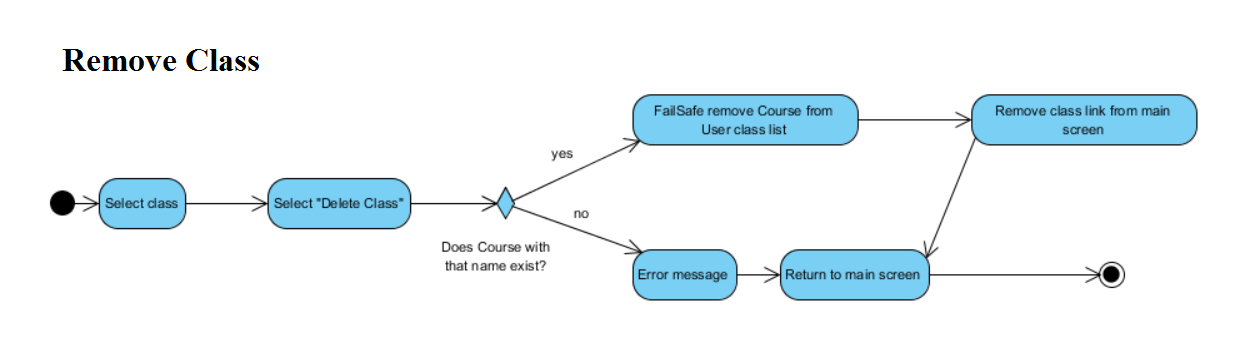
**Calculate Grade.** User selects the class for which they wish to calculate an average from their home screen. On that class’s screen they select “Calculate Grade”. The user is then presented which two options: they can calculate their actual grade as it stands at that moment or they can calculate their hypothetical grade in a “What If?” scenario.

Choose “What If?”: Failsafe displays all of the user’s current grades for all of their assignments and the user chooses which grades they wish to change. FailSafe then calculates and displays their temporary hypothetical grade.

Choose to calculate current grade: The user again is presented with two options. They can choose to calculate the average for a particular assignment set, in which case FailSafe will average the grades in that set with no weight applied. Or the user can choose to calculate their complete grade as it stands and FailSafe will average their scores with all weights and drops applied.



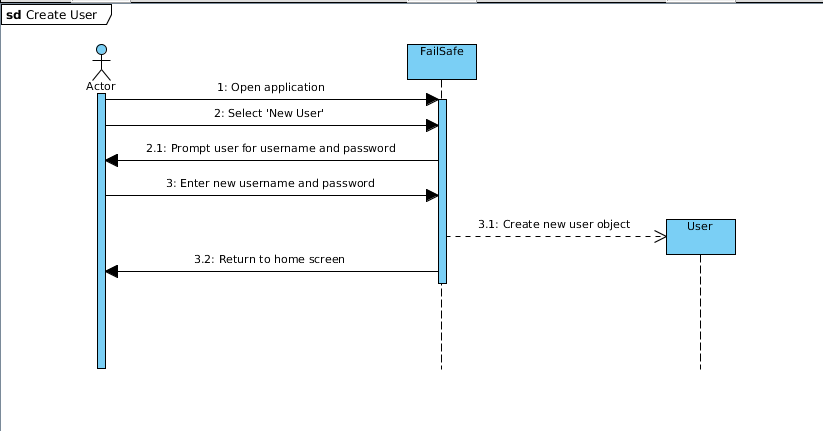
**Remove Class.** The user selects the class they wish to drop and chooses “Delete Class” from the class’s screen. FailSafe checks to see if the class actually exists in the user’s class list. If not, it gives the user an error message and returns to the main screen. If it is indeed a valid class, FailSafe removes the Course object from the user’s class list and removes its button from the main screen.



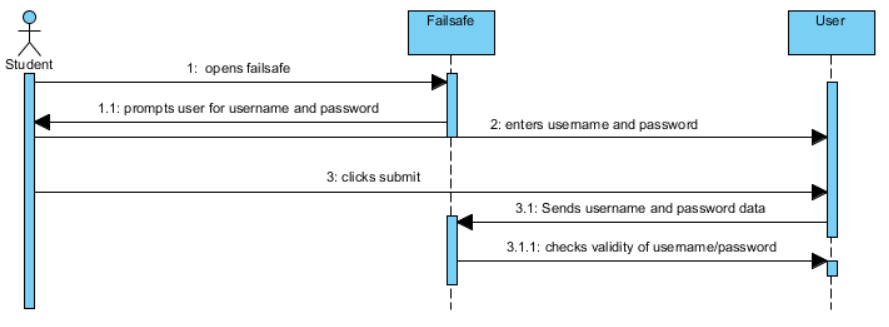
## 5.4 Sequence Diagrams

The main functionality of FailSafe can be covered in eight sequence diagrams: Create User, Log into System, Add Class, Add Assignment, Add Grade, Calculate Grade, Remove Class, Remove All Classes.

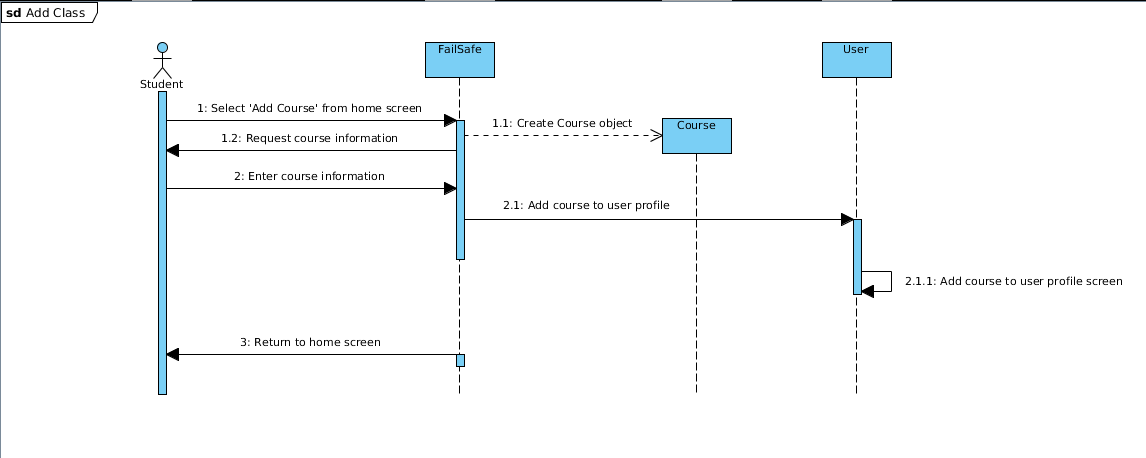
**Create User**. After opening the application, a new user can select “New User” and set up a username and password. FailSafe will then prompt the user to choose and submit a new username and password, at which point a new User object will be created to hold all information associated with the user. Finally, FailSafe will return the user to the homescreen.



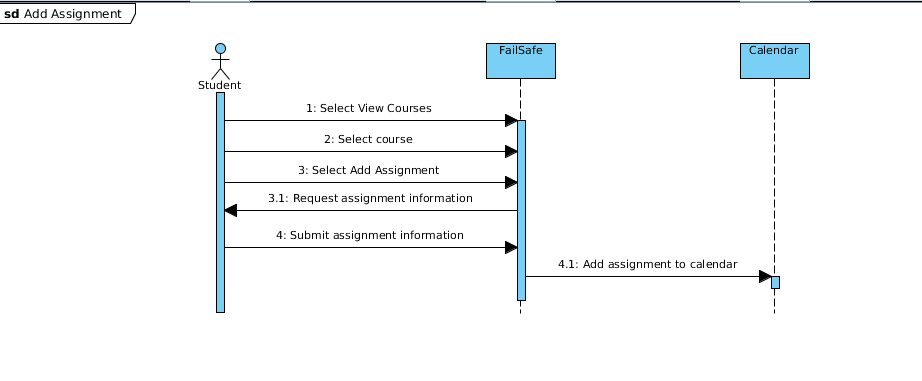
**Log Into System**. For this particular diagram the student wants to view their FailSafe profile and they must log in to do so. First student opens FailSafe . Then FailSafe prompts the user for username and password. Then the student enters the username and clicks submit and the login and password is returned to the user class. User class sends the data to FailSafe to see if it valid and here you have either valid password and the student is then taken to their perspective profile or Invalid password and the student is taken back to the home screen to try again.



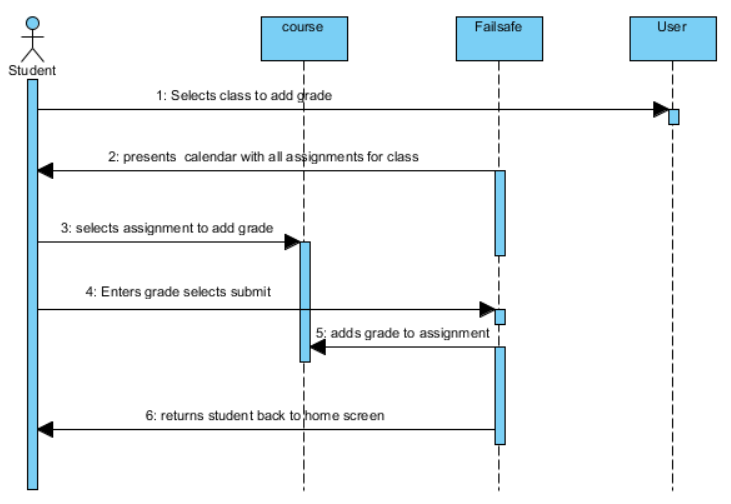
**Add Class.** To add a class, the student selects “Add Course” from the home screen and is then prompted by FailSafe for the course information, such as course name and meeting times. After receiving the necessary information from the user, a Course object is created and added to the student’s user profile. The student is then returned to the home screen.



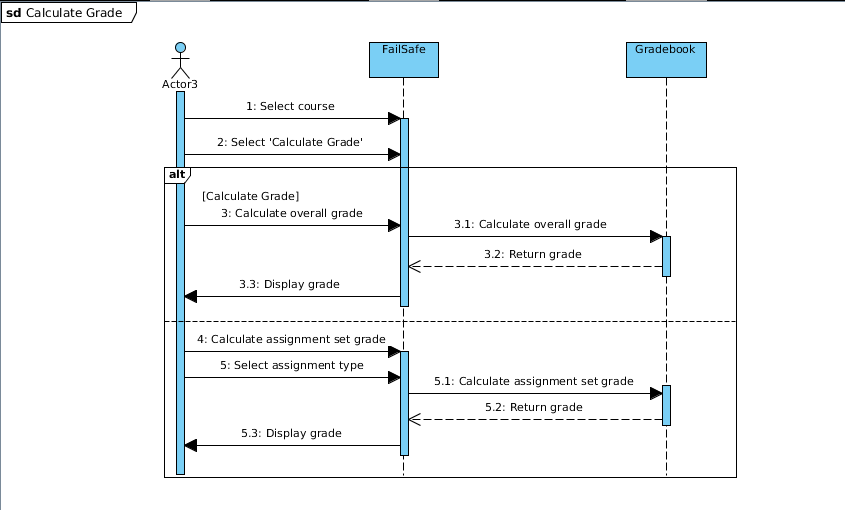
**Add Assignment.** To add an assignment, a student select “View Courses,” selects a particular course, and then selects “Add Assignment.” FailSafe will then request the assignment information from the student. After receiving the necessary information from the user, FailSafe will then add the assignment information to the calendar and FailSafe’s internal database.

****

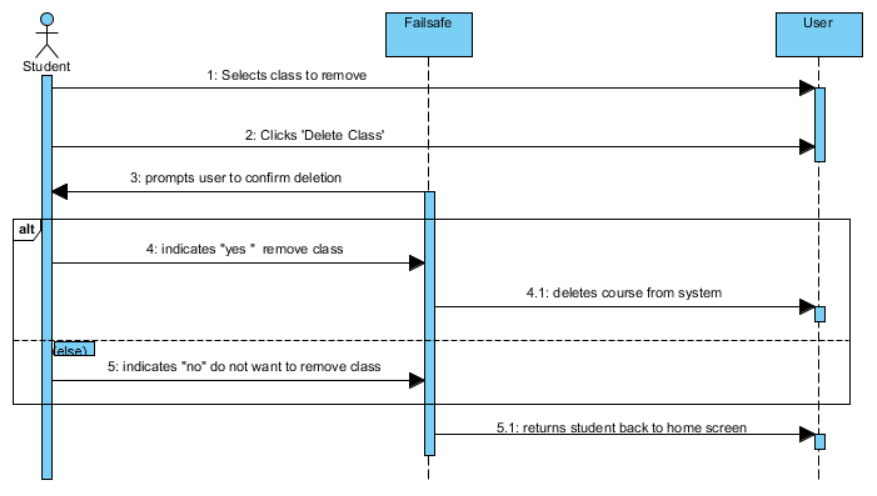
**Add Grade.** For this diagram the student wants to add a grade that has been returned to the FailSafe system. To do so the student selects the assignment they want to add a grade to from the user class. Then FailSafe presents the student with a calendar with all the assignments for a particular class. Student then selects assignment to add to grade. The student then enters the grade and selects submit and this is stored into FailSafe. FailSafe then adds grade to the assignment and returns student back to home screen.



**Calculate Grade.** To calculate a grade, whether for the class overall or just for a particular assignment set, a will select a course and then select “Calculate grade.” Afterwards, the user will be given a choice to calculate either their overall grade average or just an individual assignment set grade average, in which case the user will have to specify the type of the assignment (test, homework, labs, etc). In both cases, FailSafe will access the gradebook for the course and average the selected grades, both weighted and unweighted.

****

**Remove Class.** A student decides to delete one class from the FailSafe app. Student selects the class he/she wants to delete from the user profile. Then the student selects delete class. Failsafe then prompts the student to confirm the deletion. Here the student has two choices the first is to indicate yes remove class from here FailSafe deletes course from system and the student is brought back to the home screen. On the other hand the student can select no do not remove this course and the student is taken back to the home screen.

****

**Remove All Classes.** This function will give the user the option of deleting all existing classes stored in the application. After selecting “Remove All Courses” from the menu, FailSafe will ask the user to confirm their selection. If the user decides that they do not want to remove all of their classes, then no courses will be removed and they will be returned to the home screen. If the user does confirm that they want to remove all existing classes then FailSafe will clear any and all classes listed in the User object. The user will then be returned to the home screen.

