**Software Requirements and Design Document**

**For**

**Group AssignmentBuddy**

Version 2.0

**Authors**:

Ashley Ellis

Virginia Sicuriello

Jeffrey Knappman

Kelsey Ziemba

Analise Saunders

**1.** **Overview (5 points)**

AssignmentBuddy is going to be a learning management mobile application. AssignmentBuddy will allow students to interact with their courses, peers, and professors all through the convenience of one application. On the student side of the application, users will be able to observe their grades on assignments and see their overall standing in the class. On the professor side of the application, users will be able to create courses and assignments within the course and submit grades for each assignment. On both sides of the application, a mutable to-do list will be available, as well as a messenger to allow for a direct interface between users.

**2.** **Functional Requirements (10 points)**

We have implemented the login screen, register screen, and create account screen. We have also implemented the databases for storing the users and grades. The account screen has received additional functionality to access more pages such as their classes, a todo list, and settings. A calendar has also been implemented for the user to create and view events.

In the next iteration, we will put grade viewing at a high priority. If a student is logged in, they need to be able to view their grades in each of their classes. When clicking on a class they will be shown all their grades in the class.

The professor should be able to post assignment entries for the class and input grades. This will be where a student would view grades, a professor would be able to change them. The professor can also change the weight of each category of grade. Tests, quizzes, homework, etc. Also, in the professor’s to-do section they can see all the things the students see, as well as create reminders for themselves. High priority.

A messenger function would be available to students and professors for students to ask questions and professors to give announcements and feedback. This is a lower priority since it is not fundamental.

# **3.** **Non-functional Requirements (10 points)**

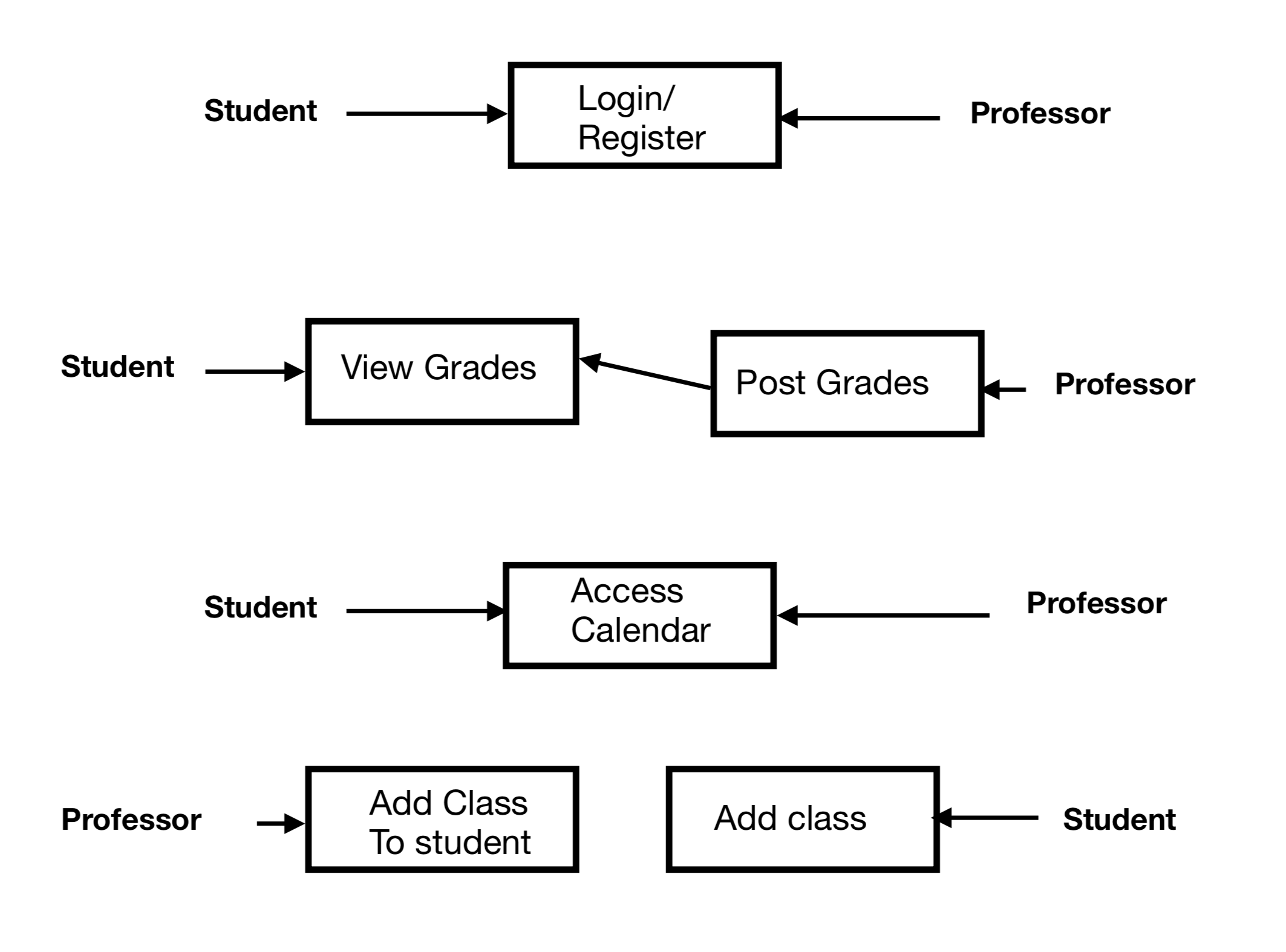
The first non-functional requirement of AssignmentBuddy is performance. Our team was always aware that multiple SQLite database tables would be utilized for the application, but we were unbeknownst to just how many we would need. As development progresses, we have found that more tables will need to exist than first calculated. Because we have to deal with multiple queries to each table within the application, performance can be slowed. So, it is important to ensure that only necessary queries are made to each database.

AssignmentBuddy requires accounts to be made for each user to ensure an individualized experience. Because AssignmentBuddy handles personal information, security is a non-functional requirement. Even though the information asked of each user is minimal, privacy is important. It is imperative that this personal information remain secured and safe.

AssignmentBuddy is a learning management system, it is vital to any university student’s life. A student’s college career is all based upon their academic standing, so it is important that our application be reliable. AssignmentBuddy needs to accurately display student academic data so students understand where they stand within a class (passing or failing). Students don’t want to use a learning management system that crashes upon opening or when examining grades, so AssignmentBuddy must be a reliable application.

Software quality is also of the utmost importance for AssignmentBuddy. We want our application to not only perform the basic requirements and functionalities, but we want the user to have a positive experience when utilizing our application. So, software quality will be a high priority for non-functional requirements.

# **4.** **Use Case Diagram (10 points)**

**

# **5.** **Class Diagram and/or Sequence Diagrams (15 points)**

Our application will be utilizing a non-object-oriented approach. The users of our programs will all be submitted into a database, so a sequence diagram will be shown to demonstrate the process of our application.

The sequence of interactions begins with the user at the application’s home screen. From here, the user can either choose to create an account or login to an existing account. If the user chooses to create an account, they proceed to the register screen. Here, the user chooses their account type (Student or Professor). After that, the user submits their information to the application. If the user has an existing account, they will proceed from the home screen to the login screen. After the credentials are verified, for both of the existing paths, the user will be taken to the account screen where all there are four options the user can pick from. The user can choose to view their courses screen, view their calendar, go to the application’s settings, or view their to-do list. At any state in the sequence, the user has the opportunity to back track to the previous state.



# **6.** **Operating Environment (5 points)**

Our software is an android app. It will run on android hardware and android 4. We are testing it on a simulator of a Pixel 3.

# **7.** **Assumptions and Dependencies (5 points)**

This application utilizes Android Databases’ SQLite API. An assumption that can be made with regard to it is that the library will securely store the information of our users. It is also a dependency, because without this API our application will not be able to store all of the necessary information required for our users.

For our development environment, we are using Android Studio and will be using the emulator to run and test the application. Sometimes there are bugs within the emulator even though the code may not have errors. So, an assumption is that the emulator will run and execute the code properly.

In the past, Ashley has done projects that utilized the SQLite Content Provider in Android Studio. So, as a dependency she will model the database creation, entry insertion, entry deletion, and entry updates after the way she implemented it within that mobile programming project. Another dependency is the code implemented to create notifications within the application’s broadcast receiver. Ashley modeled the Notification/Notification Manager code from code implemented in her past projects.