**Software Requirements and Design Document**

**For**

**Group AssignmentBuddy**

Version 3.0

**Authors**:

Ashley Ellis

Virginia Sicuriello

Jeffrey Knappman

Kelsey Ziemba

Analise Saunders

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

AssignmentBuddy is a learning management mobile application. AssignmentBuddy allows 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 for assignments and analyze their overall class standing in the based on the calculated average. 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)**

Upon first opening the app, users are presented with two options, login or register. When clicking on login, the user can simply login. When clicking on registers, they are presented with the choice between student or professor, so we have a difference in the accounts. When they login, they see an account screen with buttons for a calendar, todo list, settings, messenger, and courses. People can use the messenger to send messages, and professors can use it to send reminders or post announcements. The calendar can add dates on a specific date and time. There is also a way to create tasks in a to-do list, so students and professors can remind themselves of assignments or tests. In settings you can change your password and logout. They can view courses by clicking on the courses button, and it would take them to a list of their courses, and there they can add a course. By clicking on a course, it will show all the grades for that course.

# **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. It’s also crucial that our team identify potential vulnerabilities within our application - specifically, username and password protection, most of which is verified through our database. Our team must also consider abuse of the application - in other words, assess several misuse cases in which a user could exploit certain aspects of our application.

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.

# 

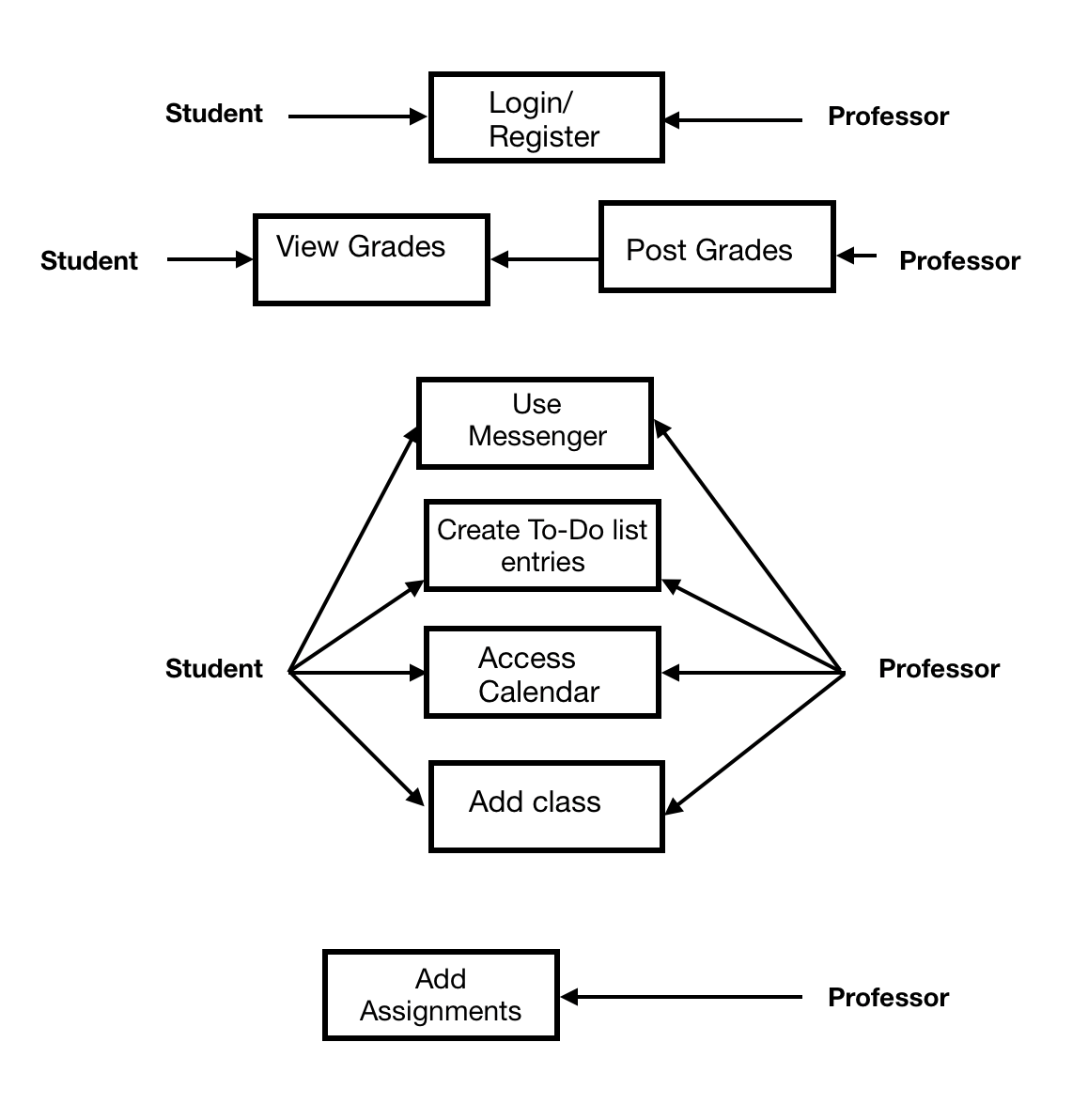
# 

# 

# 

# 

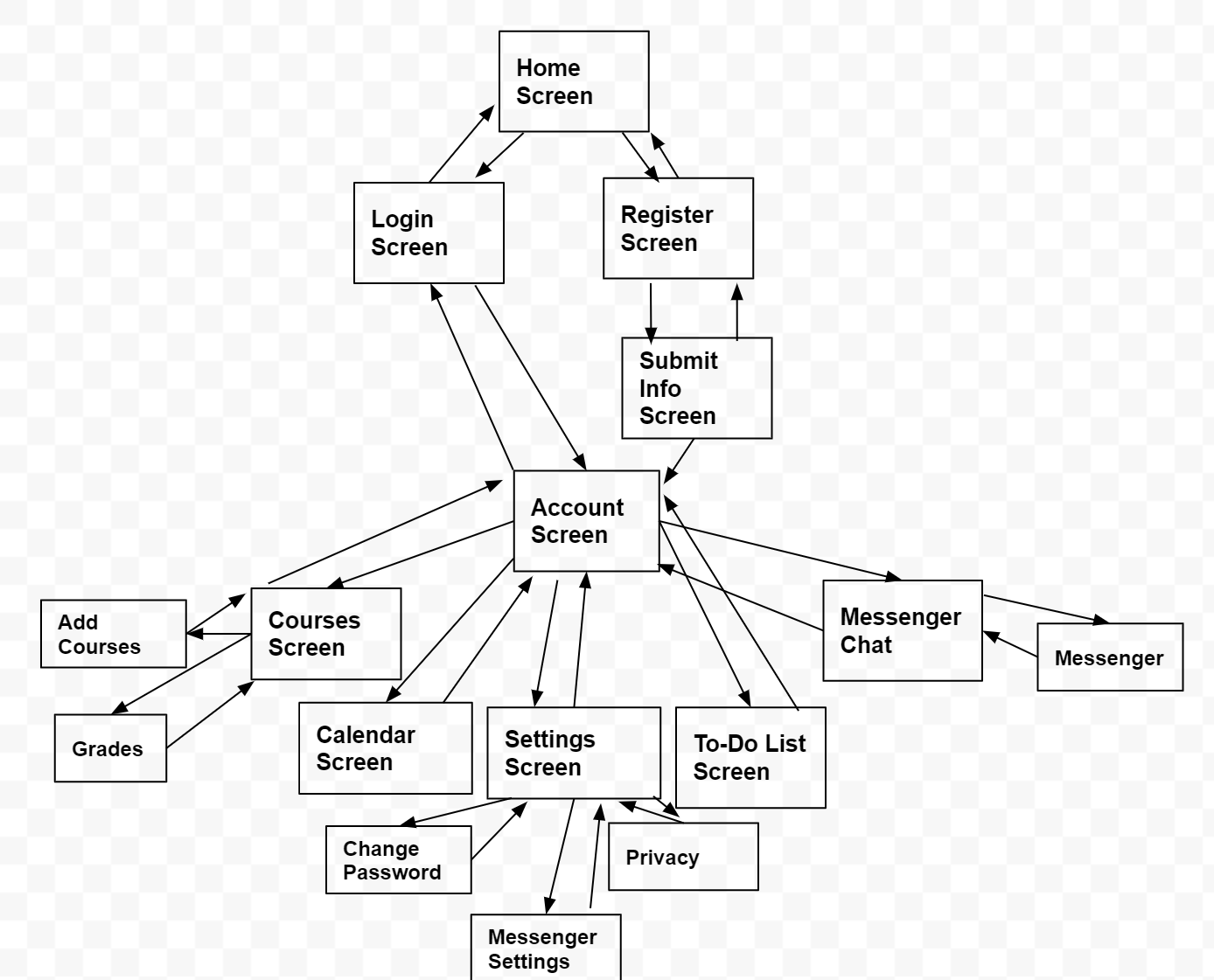
# **4.** **Use Case Diagram (10 points)**

**

# **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 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. Implementations made throughout Iteration 3 extended the sequence diagram’s number of states. At any state in the sequence, the user has the ability to traverse 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.

A newly added dependency was implemented during iteration 3. This dependency was for the messenger activity. Ashley researched existing projects that implemented similar features to create the design and code for modeling a messenger-like activity.