**Software Implementation and Testing Document**

**For**

**Group AssignmentBuddy**

Version 2.0

**Authors**:

Ashley Ellis

Jeffrey Knappman

Analise Saunders

Virginia Sicuriello

**Note:for the first increment, only sections 1 and 2 of the document are required.**

**1.** **Programming Languages (5 points)**

The programming languages used throughout the project are **Java**, **XML,** and **SQL**.

In our project, **Java** will be used to invoke activities within our Android application. An activity provides the window in which our app will draw its UI; one activity implements one screen in an app. Developing a .java file is necessary for developing activities within our application. Our group chose to use Java because it is the language primarily used in Android development and it is supported by Android Studio (IDE where we are building our app on). Java utilizes object-oriented design, which all members of our group are familiar with; and it contains many frameworks and class features that are beneficial in the development of our project.

**XML** will be used in conjunction with java to implement activities within our app. Currently, XML is being used to store data about our activity - such as activity size, layout, etc. Since XML makes data storage easier, we may use it to store necessary user information. We chose XML because it is a necessary aspect in developing the UI for our app, and it contains several predefined widgets in android that would aid in our project’s development.

**SQL** will be used as a means to manage data that we would want to store in our application. SQL would primarily be used in overseeing data management of user login information and other necessary user data that relate to the user’s personal data, such as, university courses that would be stored within the app. We chose SQL because it's great for querying data - which is important. After all, our app is going to rely on quite a bit of data storage. It can also be used in conjunction with XML.

# **2.** **Platforms, APIs, Databases, and other technologies used (5 points)**

The platform for the project is **Android Studio**, it is the IDE where our app is being developed on. The emulator that our application is running on is **Pixel 3 API 15**. We will use this emulator due to this preloaded hardware profile - this relates primarily to the applications UI. The API of the emulator that is being used to run our application affects how our XML files will be developed. Android Studio provides a template-based setting which makes it easier to visualize how the interface will look like. Further into development, our group plans on using **SQLite API** in Java in order to create and maintain database tables within our application. SQLite would primarily be used in relation to user login information, including data concerning registered courses and assignments. It will also be used to help store the classes pertaining to each user, and the grades within each class and assignment.

**3. Execution-based Functional Testing (10 points)**

Unit testing will be the functional testing used for this project. Each member has their personal branch in order to test the part of the software they are working on, this way each unit is tested separately checking if it is performing as expected. The group decided that only the units that are working correctly will be uploaded to the master branch therefore, this branch will have the latest version of the code without errors. In addition, the emulator is also used as a visual way to check that everything is working correctly, and the layouts are centered. Before every increment, each member checks all units to attribute in them or help if needed.

Additionally, with the implementation of a few databases, our group members can now make data entries into our application and test whether any of the user’s input could possibly ‘break’ our application - this also tests whether our database’s relational design has been implemented well and effectively. For example, most of our application’s calendar has been made functional this iteration for testing. Therefore, our members have tested its event insertion functionality by trying to add an event to the calendar in several different ways made possible by our apps UI. This method of testing can tell us if our code implementation is robust and reliable.

**4. Execution-based Non-Functional Testing (10 points)**

During this iteration, our team has begun implementing several databases as a means to manage much of the user’s information. A database has been created for the user’s login information, as well as their grades, courses, and calendar. As a result, much of the **execution-based testing** regarding our **non-functional requirements** has been on **performance**. A primary concern has been on whether our application’s performance speed has been compromised with the usage of so many databases. Fortunately, we have yet to be dealing with large amounts of data, but it is certainly an aspect we pay close attention to when conducting an execution-based test on our app. For example, during this iteration, we paid close attention to the speed at which our Calendar function has been able to perform event insertions and outputs. This is primarily done by running the Android’s phone emulator and conducting a few event insertions into the Calendar to see whether the app’s functionality has been slowed down with each data entry. For now this does not seem to be an issue, but it can be when more data is funneled through our application.

The group has also manually checked all the code to ensure the correct data is used. In addition, using the case diagram is key to visually see and make sure the data used in every unit is the expected one. Also, reviewing the effectiveness of the code in order to make the application as fast as possible. In the future, account security will be implemented since it is imperative to maintain the information of the users safely.

**5. Non-Execution-based Testing (10 points)**

Much of the **non-execution-based testing** occurs towards the end of our iteration - as each team member pieces their code implementations together. Small code inspections are done individually as members prepare to merge their branch into master - ensuring implementation conflicts are resolved before its commit.

Towards the end of our iteration deadline, each member is required to submit a video screen and audio recording of their code and demo walkthrough of the functionalities they’ve implemented from the time of the last iteration. This video and audio is condensed and used for our iteration video submission - **however**, it has provided our team a great strategy to conduct a smaller scaled version of a non-execution based test. Each member spends time reviewing each member’s demo walkthrough while following along to the project’s proposal and RD document - outlining functional and non-functional specifications. It's recommended that each member takes time to review the code for themselves to provide a more-well rounded assessment. In our follow up meeting, we provide feedback and voice any concerns we may have come across during our code inspections. Scheduling a meeting at a later date, after the code reviews, provides each member ample time to prepare comments and questions.