[[1]](#footnote-1)

A Technical Overview of “Calculot”

Kenneth Zheng, Levent Kylych, Ryan Peterson, Jean Pierre Acebedo  
Simon Fraser University  
British Columbia, Canada  
[kza21, lkylych, repeters, jacebedo] @sfu.ca

*Abstract*— This document focuses on the software development of Group 14’s application, “Calculot.” In retrospective, this document will focus on the techniques, software packages used, constraints and limitations of the development of Calculot. As this is a simple application consisting of a game with simple learning topics, the technicality involved in the document will be considered as a document that concerns a high-level understanding of software development and implementation techniques.

# INTRODUCTION

# background

<INSERT TEXT HERE>

# Techniques Used

## Databases

Storing information after the application, “Calculot”, closes was deemed an important part of the application when deciding functional requirements. The user must be able to store information about their progress after the application has been closed. More so, in the case that one would choose to share the application between multiple users on the same device, the application must be able to support multiple users and can keep track of each individual user’s progress separately.

SQLite was chosen to be used as the database manager for “Calculot” for various reasons. The development team for the application was already familiar with the implementation of SQLite in an Android application from past experiences. Moreover, the manager is natively supported by Android [6], and features a relatively simple interface that allows its implementation to be done without problems.

After beginning the application and not already logged in, the user is asked to sign up within the app. This sign-up activity extracts the inputted user information and stores it in the SQLite database, creating the user’s account. Information asked for by the sign-up activity includes: chosen username, user’s first name, and the user’s chosen password. The password field must be entered twice to ensure that it is correct.

The user also can add multiple users, one at a time. The same fields are required to be filled, and the user information is stored in the next row of the database.

Throughout the application the user gains experience points (“XP”), based on their progress through the learning activities and game activities. These are also stored in the database respectively as “Learning XP”, and “Practice XP”.

The information described makes up the user’s profile within the app and is called upon by the application to show their progress through the application.

## SharedPreferences

During the first sprint of production, the development team decided the application must keep the user logged in until they decide to log out. The application must also keep the user logged in when the application is closed.

The best approach to this problem was decided to be using SharedPreferences. SharedPreferences is a common method used to store data in an Android application even after the application ceases. SharedPreferences has been include in Android since API level 1. It allows the developers to store a piece of data with a string (“key”) that is used to retrieve it [7].

After the user selects their username from the list of registered users, and then enters their password correctly, they are then considered logged in. The application implements this by storing that user’s username in SharedPreferences under the key “username”.

Upon starting the application, SharedPreferences is checked to determine if a use is already logged in. If the string pulled from the SharedPreferences is null, then the application’s main menu will ask the user if they would like to sign up or sign in. If the string pulled from the SharedPreferences is not null, then the application’s main menu will instead show a different set of options.

SharedPreferences are used throughout the app to determine information about the currently logged in user. If the development team required any user information besides the username, they are required to call from the user database for the rest of the information. The implemented database returns a custom “User” class that holds all the user’s information. From here the developer can use any information about the user wherever that User instance carries on.

Because the user is kept logged in to their account even when closed, a way is needed to allow users to log out of their account and switch users if needed. When the user is logged in, the main menu displays a button to log out. Upon pressing the button the “username” field in SharedPreferences is set to null and the menu is set to the menu that appears when not logged in, thus logging the user out.

# Constraints and Limitations

## Preface

Due to the simplicity of the application and the unfamiliarity of android and java development of the team, “Calculot” as an example of a standardized application proved to be constrained and limited to simple design and implementation techniques. The issue can be generalized as an issue of understandability vs. performance gains of the games. In short, the games had to be as understandable as possible while also providing a playable experience to the user. As such, the following points of the development of “Calculot” was constrained:

1. The limited range generated by the question generation function
2. The notation provided in “Calculot” was done using a simpler, but more effective alternative,
3. Varying systems can differ in entertainment experience and difficulty.
4. Simplicity in design prevents complexity in development.

## The random question generation

Since “Calculot” is based on low-level university courses, the topics must be simple enough to provide entertainment to the user along with providing an effective learning schema to help the user understand the topics involved. Because of this, the question generation for the games implemented in “Calculot” had to be limited in design to prevent unreasonable questions and to create a fair experience for the user. The generation of the questions limited in the format and the range of constants to provide a solution to this issue.

Table I

Formatting of differentiation question set of “Calculot” in easy difficulty

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Question | Answer 1 | Answer 2 | Answer 3 | Answer 4 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

As shown in Table 1, the formatting of “Calculot” had to be limited to simpler questions based on the topics. However, the answer set provided extra difficulty through making the questions somewhat similar to the solution string. This formatting is designed to punish the user from being unaware and unfocused on the question while also providing a fast-paced and interactive experience.

Similarly, the range of the constants and exponents in “Calculot” were limited reasonable integer values. This is to allow a reasonable pace on integrating or differentiating the functions. Allowing for unreasonable multiplication can result in a detrimental learning experience for the user as the objective of the game from a motivational standpoint is to allow the users a platform to display and/or practice their skills in advanced mathematics. Struggling with mental arithmetic should not be the focus of “Calculot” and should be treated as a trivial step to the solution.

## Notation

The notation in “Calculot” is an issue of android compatibility. Mathematical notation techniques have not yet been implemented and is currently a brute-force approach for mathematical applications. A workaround this issue would be to incorporate the android WebView [1] and HTML math notation using MathJax or KateX to the application. However, the drawback is a significant decrease in responsiveness and performance in the “Calculot” games. As such, the implementation of mathematical notation is to use a classical notation, while using constants. The implementation of “Calculot” used common ASCII symbols, such as “/” and “^” to incorporate the mathematical functions, such as

= x^n

= (k/l)

The decision to use a classical notation is motivated by prioritizing performance over understandability. While the understandability of the mathematical functions is severely affected by the notation, the performance drawback caused by WebViews caused both games in “Calculot” to be near-unplayable.

## Difference in platforms

The smartphone advancements in the past decade have advanced substantially. With new graphics and computer hardware becoming faster and faster, the experience of different applications can change dramatically, when comparing two extremes. The same reasoning can also apply to “Calculot.” The way that the integrated android Canvas class [4] operates is that the class calls the method onDraw() as much as it can, which allows the class to draw each object to the screen. The rate at which onDraw() is called can vary based on the device specifications; some devices will call onDraw() much faster than other devices. This can result in the game providing slight differences in the form of the enemy’s velocity as it progresses toward the tower. A possible solution to this issue is to limit the calls of onDraw() to a minimum. Equivalently, this will lock the frame rate of the graphics for “Calculot” However, this reduces efficiency further and older devices will be impacted more heavily than the benefit for newer devices. The development design focuses on simpler game design and the development team decided to broaden the range of compatible devices that “Calculot” can support.

Similarly, the screen resolution of different devices proved to be an issue for “Calculot.” More specifically, the sizes of each game object differed in size for different screen resolutions and the velocity of the monster was also impacted by screen resolution for “the Defense of Calculot.” Both of these issues were solved through scaling bitmaps and using mathematics to place and move each object based on the screen resolution. An example of a mathematical functions that were used were using this format:

Where the dimension of the object was the height and width, separately. The reasonable % of screen was achieved through trial and error, where the graphics were drawn on one mobile device and tested for a plethora of other devices.

In conclusion, the experience of “Calculot” can vary depending on each device, as varying devices have different screen resolution, CPU speed and other hardware. The development team behind “Calculot” attempted to minimize the impact of the differences in media through using workaround methods in the application and as a result, provided somewhat comparable results across a range of devices.

## Simplicity in Design and how it impacts current and future development

The way that “Calculot” was implemented and designed is an straightforward design compared to other applications of the same domain. This can prove to be an issue as some users can find “Calculot” to be too simple in nature, and not have an enjoyable experience as other might have. However, this opinion can also vary as some prefer the simple entertainment method to also help them learn mathematics.

There are also other limitations that is a result of this straightforward design. As mentioned previously, the issue with compatibility of a range of devices can be attributed to the design of “Calculot.” The implementation of the workarounds proves to be a difficult issue due to this. More specifically, the frame rate lock for “the Defense of Calculot” becomes a trivial issue in terms of implementation as the experience gains for this came is minimal compared to the impact it will have on older devices.

Another example is the decision to use the SQLite Database [5] While SQLite provides a very effective method of storing organized data, this only stores the data locally, and can provide issues later: the size of the database might grow to affect the user’s storage as it grows to a large size, or the issue that the user has no way to back up their user data. Due to the decision to use SQLite for the simplicity and effectiveness of a local database, the modern standard for entertainment-based applications is a cloud-based online webserver, which the user can back-up and maintain through the web.

Lastly, the incorporated Canvas class that “Calculot” uses operates in a simple manner. When the onDraw() is called, it simply displays all of the objects at given coordinates and sizes. A much more effective way of providing graphics for “Calculot” is to use an integrated game engine, such as unity. However, due to the unfamiliarity with these alternative methods, “Calculot” had to be programmed using the OpenGL Canvas class for its simplicity.

Though there are many constraints to the simplicity in design, it also provides many benefits. It helps to gauge all of the advantages and disadvantages to decide which method will deliver the best experience to a broad range of users.

# 

# 

References

[1] <https://developer.android.com/reference/android/webkit/WebView.html>

[2] <https://www.mathjax.org/>

[3] <https://github.com/Khan/KaTeX>

[4] <https://developer.android.com/reference/android/graphics/Canvas.html>

[5] <https://www.sqlite.org/>

[6] <https://developer.android.com/training/basics/data-storage/databases.html>

[7] https://developer.android.com/reference/android/content/SharedPreferences.html

1. [↑](#footnote-ref-1)