# Project

# Introduction

In 2017, there were approximately 600,000 students taking maths and science GCSE exams. In maths, 31% of these got the equivalent of a 3 (equivalent to a D) or below. There are many programs and apps that solve the problem of having a portable calculator, however, I believe that they could be improved upon to better help students.

In 2014, there were a total of 288,000 students entered for STEM A Levels. Approximately 25% of those students exited Sixth Form with a D grade or below. I think that the difficult mathematics that are included in STEM subjects are a large contributing factor to those who get lower grades. Therefore, I think developing an app that will cater specifically to students will help them improve.

My project aims to solve this problem by creating a scientific calculator Android app that is specifically designed for GCSE students, so they can have access to better tools to enable them to achieve their best possible grades.

# Analysis

## Features and Computational Amenability

The scientific calculator I aim to make will have a multitude of features that will be well suited to being created using a computational approach.

One such feature is that the calculator must be able to perform basic arithmetic operations following the rules of BODMAS. This feature is amenable to a computational approach because using the mathematical operations can be calculated much quicker and with much more reliability compared to the traditional methods of using a pen and paper which can be very prone to mistakes.

Another feature important to this project is to have the calculator in an Android app because it provides a degree of portability and ease of use that is very important for the app to be successful. This feature lends itself well to computational methods of solving because using a computer is the only way to provide an Android app to the users.

This project will also have the capability of having some common maths and science equations loaded where the students can input their variables into the calculator to solve the equation to help students as they are revising to check their answers. These inbuilt equations will also contain walkthroughs to manually solving the equation and revision tips.

This is amenable to being solved with a computer because the average phone has plenty of storage to hold all the information required to store different equations. It’s suited to a computer because it is compact and should be intuitive and easy to use, unlike traditional methods such as keeping the information stored in, for example, a revision book because they are heavy and sometimes difficult to navigate. This makes my approach of using an app beneficial to the user because it streamlines their process of work and revision.

I also hope, if possible to have some basic graph functionality where the user can enter linear, quadratic and cubic equations into the app and the user will be provided with a graphical representation of that equation. This feature is important and will be solved easier with a computer because drawing one with a pencil will take much longer and is more vulnerable to errors.

### Stakeholders

This project has multiple stakeholders which are listed below with their requirements:

1. **Students;** specifically, they will be students in Year 10, 11, 12, and 13 currently taking the GCSEs and A Levels. They will make up the majority of my potential user base. This stakeholder will have a high interest in this product because it will be designed specifically for them.

Requirements:

1. This stakeholder will have the main requirement of an Android phone with at least 2GB of storage and at least 3GB of RAM so they can store and run the program.
2. Another requirement for the students is the compatible Android version so the app will run on their phones. I will be developing this program to be compatible with either Marshmallow or Lollipop as these are the most popular versions with 32% and 27.7% usage share respectively as of October 2017. This will give the project the largest potential user base possible
3. Another important requirement is the correct aspect ratio so that the program will be presented correctly and be properly functional.
4. **Teachers;** these are another important stakeholder because the teachers must be aware and knowledgeable about the program so that they can introduce it to the students.

Requirements:

1. The main requirement for teachers to have is a phone with the correct version, either 5.X or 6.X. This will enable them to access the program as it will not be compatible with less popular versions as that would take too much time to program.
2. Another requirement for the teachers would be for their phone to have enough RAM and storage space to be able to run. However, it is unlikely for the program to take up a lot therefore, an average phone should have more than the capacity to run the program.
3. Also, teachers will need their phone to have the correct screen resolution, so the app is displayed properly on the device.
4. Another important requirement is an internet connection to enable the stakeholders to download the app from where it is hosted, for example, the Google Play Store.
5. **Parents;** this is potentially the least sizable stakeholder but are still important because the parents will need to have access to the app because it will enable them to help their children with their schoolwork where they may not be able to otherwise.

Requirements:

1. A phone with a minimum of 2GB RAM and 4GB storage to enable the stakeholder to be able to use the app properly.
2. Their phone would also need to have a compatible screen resolution that will provide the display for the user that makes the app easily readable.
3. Another requirement is an internet connection, so parents can download the app from where it’s hosted, and also download updates if they are made to ensure that their app is the most recent version possible.
4. The parents will also need their phone to have the correct version, so the app is compatible with their phone.

## Research of Existing Solutions

During my research of this problem, I have found many different solutions that solve the problem, or at least partially solve it. The various solutions achieve different levels of success and therefore, I will be analysing different apps.

### 1.3.1 HiPER Scientific Calculator

This is the first example of a very successful solution that I came upon. The HiPER Calculator is presented in the traditional scientific calculator form and is very easy to interpret.

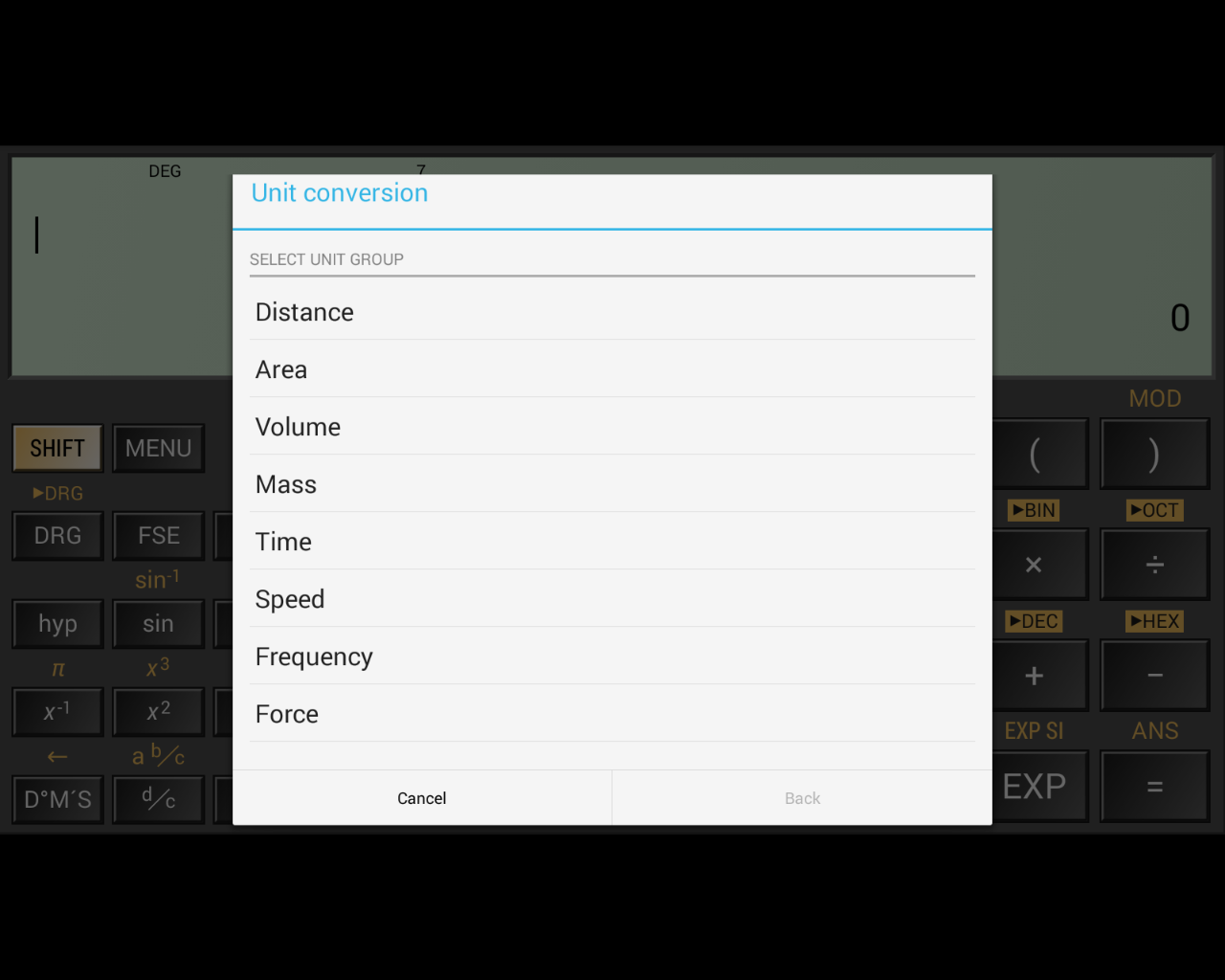


*The basic layout of the calculator*

This app has many positive points about it. For example, it’s usage of the traditional layout means that the users will instantly be able to interact easily with the app.

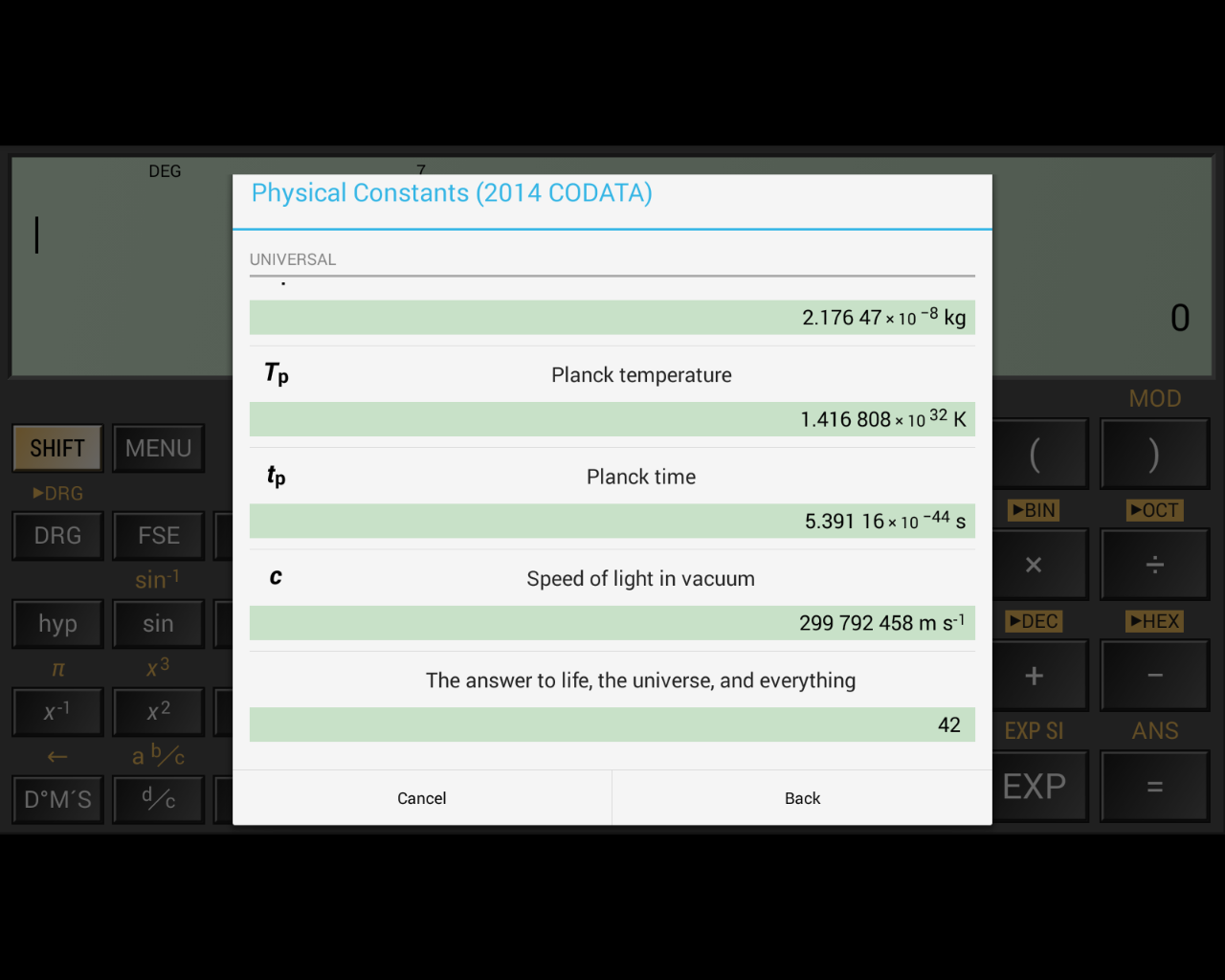
The main positive point about this app is that it performs very well and is responsive to user inputs which means that it’s quick and easy to be used. This app covers all the basic features and functions such as conforming to BODMASS, basic trigonometry, surds, powers, etc. However, the calculator also contains inordinate amounts of other, more niche features that make this calculator much better than the rest on the market.

An example of this is the large amount of hidden functionality that it contains behind different menus. For example, it has the ability to convert between almost any units:



*The different possible conversions*

Other features like this include the storage of common constants that are used in scientific fields.



*Different possible constants – there are many more*

This feature is incredibly useful for students as it reduces the amount of time it takes to complete calculations which can be especially useful in certain situations, for example tests. I will be almost certainly adopting this feature in my own project because I believe that this improves the usability of the calculator a lot.

However, I will make my own changes to this because as my own calculator will be specialised towards students in the years. Because of this, I will remove some of the constants as they will not be used up to GCSE and therefore will be useless.

Also, I will be altering the constants to fit in line with the exam’s own values. For example, the speed of light in vacuum is 299,792,458ms-1, but in exams the value is taken to be 3.0x108. Therefore, it is counterproductive to GCSE students because they will receive a slightly incorrect value which could result in rounding errors in their calculation, leading to a loss of marks.

However, this app does also come with a relevant flaw in that the user cannot store numbers as variables for ease of use. The memory storage in this app is reduced to M, and 0-9:



*The calculator’s storage capacity*

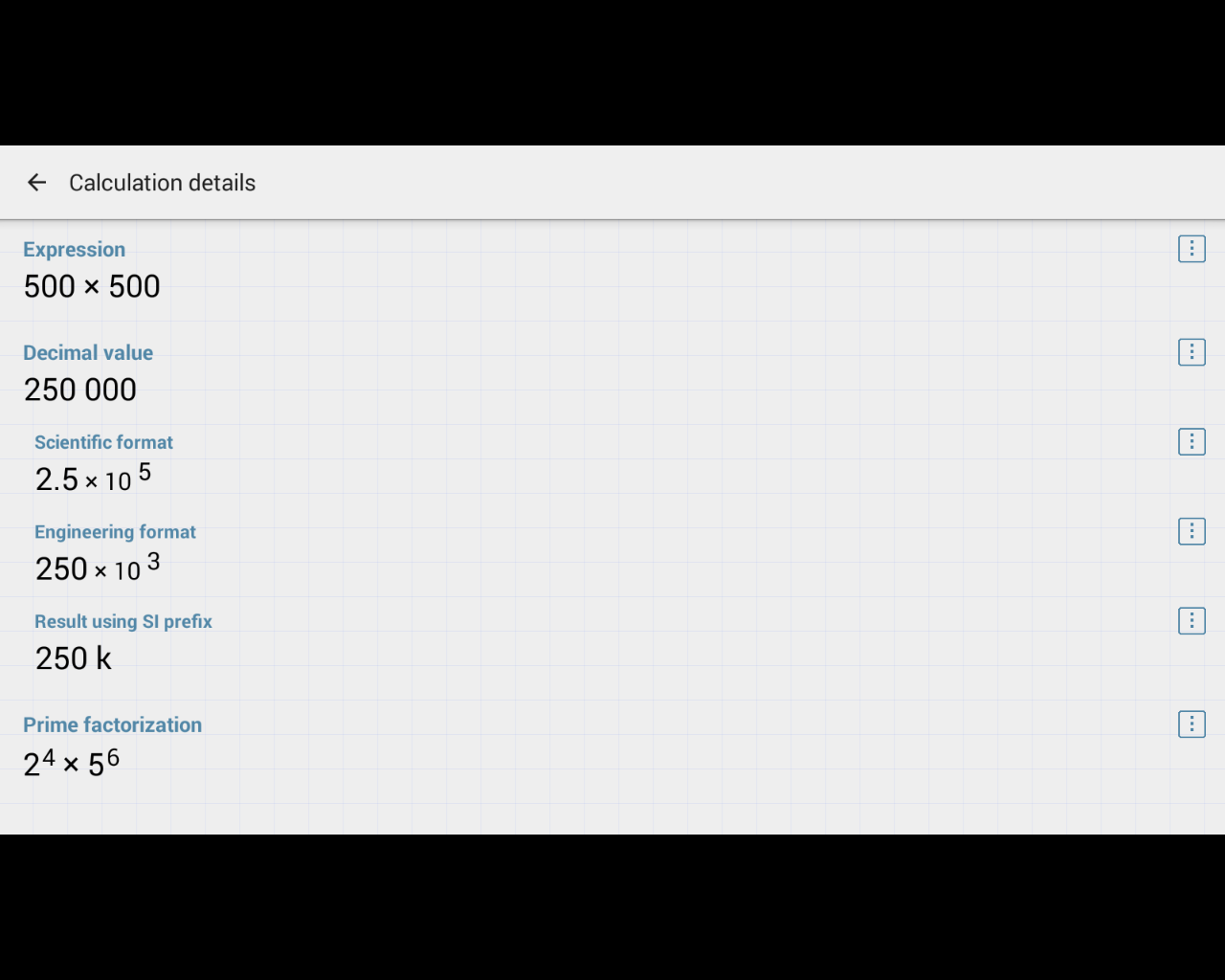
I believe this is a major flaw in the calculator because the ability for the user to store their own preferred values in variables is very useful to make the use of the calculator more efficient. Also, the added customisability it provides means using the app becomes more intuitive.

The HiPER Calculator also has the functionality to convert numbers between different representations, such as binary, hex, and denary. I think this feature can be very useful, especially to particular students, such as ones taking Computer Science. This feature is further extended past just simple conversions by having modes for different base number systems.



*The calculator in the binary mode*

As the screenshot shows, 255 has been converted to 1111111, and the app has entered the ‘BIN’ mode where the features are different. Prominent examples of this are all the keys on the number pad except ‘1’ and ‘0’ are disabled. The letters from A – F are also enabled when ‘HEX’ mode is enabled. Further features include the inclusion of logical operators such as ‘AND’, ‘OR’, etc. which would be very useful for the user when performing binary calculations.

Other helpful features include things like the ability to copy the output to the clipboard by tapping on the number. The app also has the functionality to display in depth details about the calculations the user is performing. 

*The output of the ‘More’ button*

As shown in the above screenshot, by clicking on ‘more’, the user is presented with further details about the calculation which involves things like displaying the result in standard form or with different prefixes.

These features and other ones like them are things I would like to include in my final project, however, they are considerably less important than other aspects of the app because while they are helpful, I feel that they are not as necessary to the calculators overall functionality. There are also some features that are useful, but not necessary in my own project, such as the functionality for complex numbers because they are not introduced until A Level and therefore would be pointless in my project.

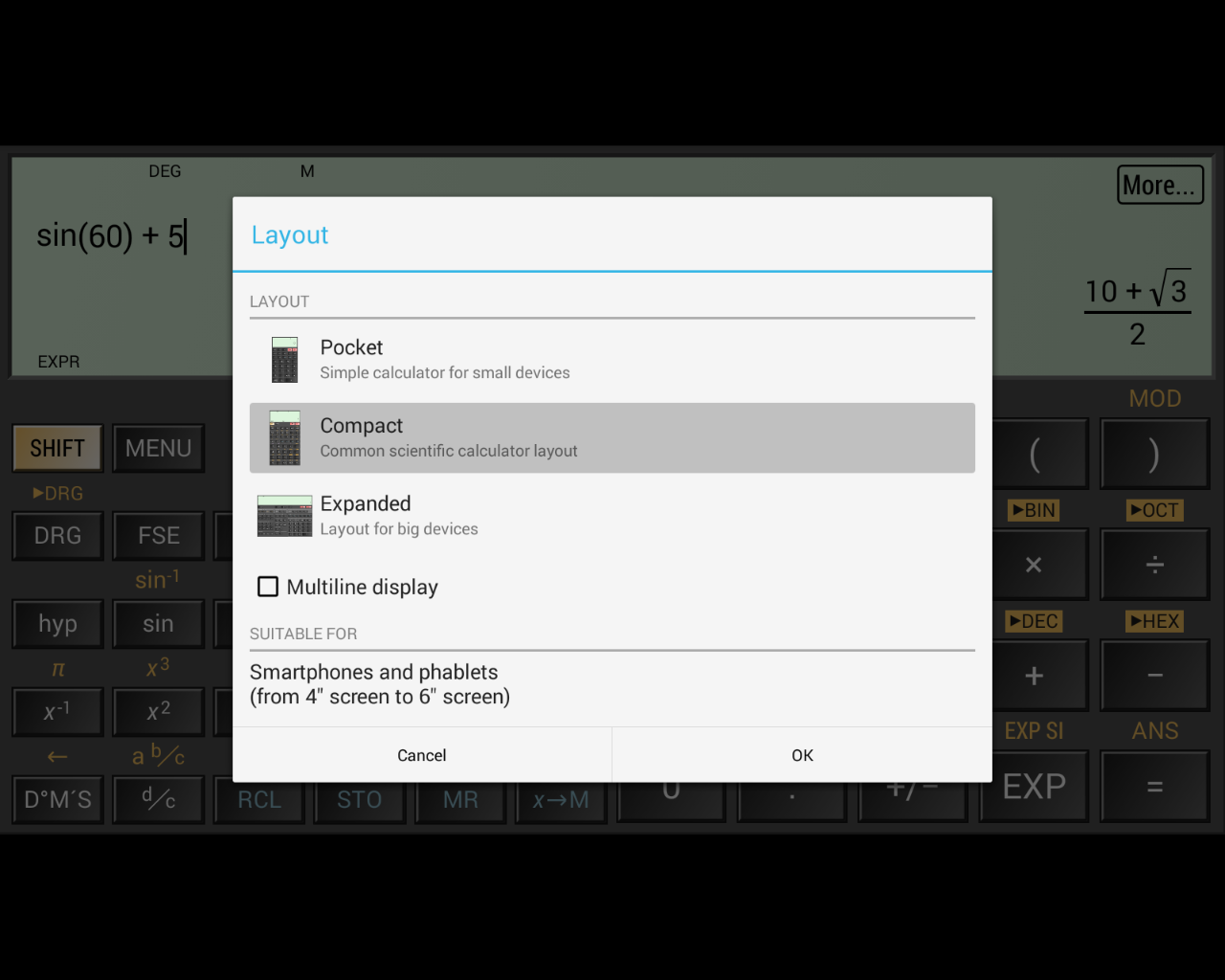
This calculator also has many benefits other than it’s many features, for example, the app itself is presented in a very clean and uniform way that makes it enjoyable to use as opposed to others on the market which can tend to be gaudy and unprofessional. A good example of this is the app’s menus:



*The calculator displaying different formats for the result.*

In my opinion, the presentation of the app and its different menus is very sleek and minimalist which makes using the app easy and intuitive.

There are also other benefits to using this app as opposed to others relating to it’s presentation. For example, there are choices in the menu for different layouts. This is an important feature because there is a risk for users with uncommon device sizes to have the app be presented with a distorted view that can block off some buttons and features that limit the apps usability.

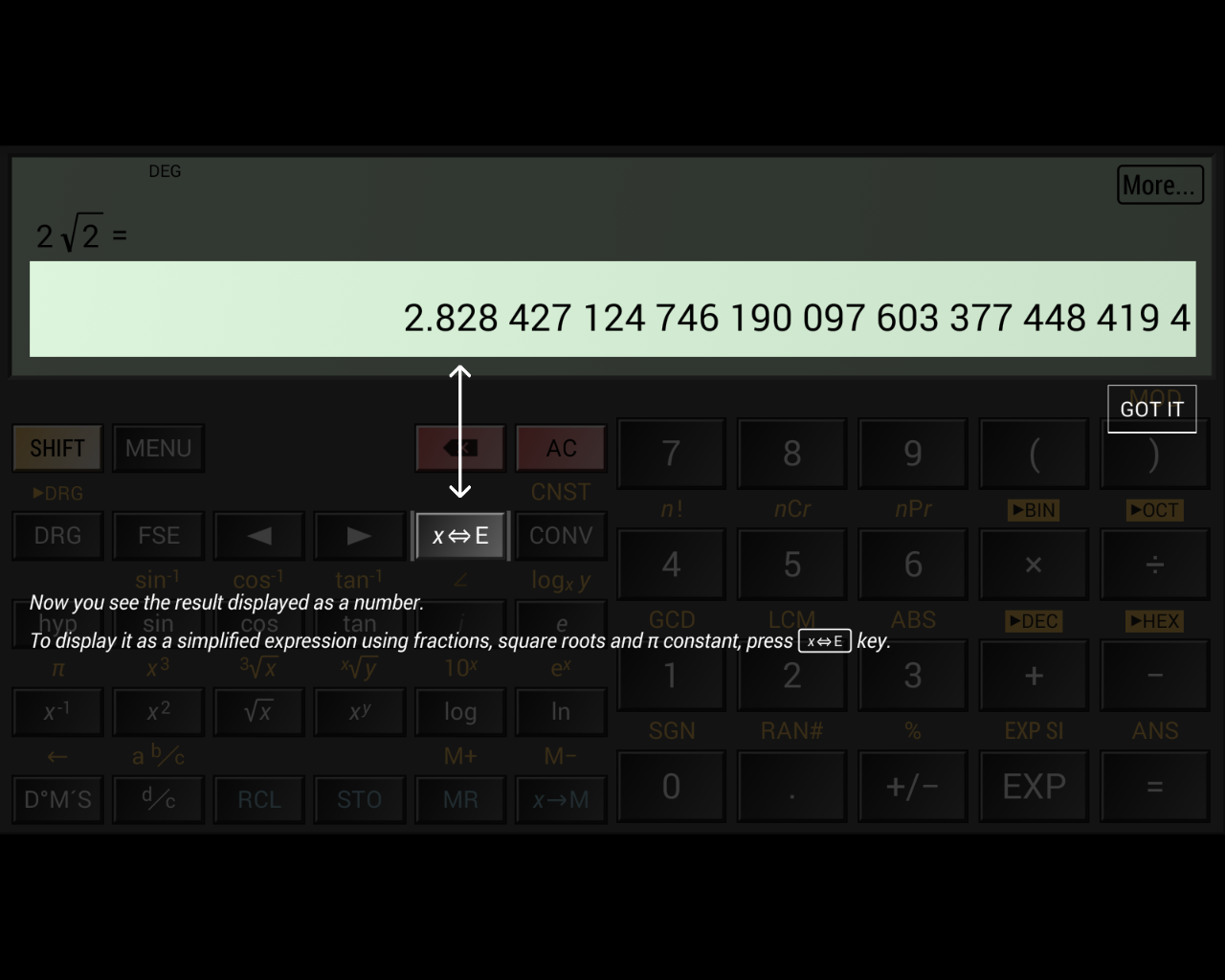


*The different layout options*



*The layout mode for tablets*

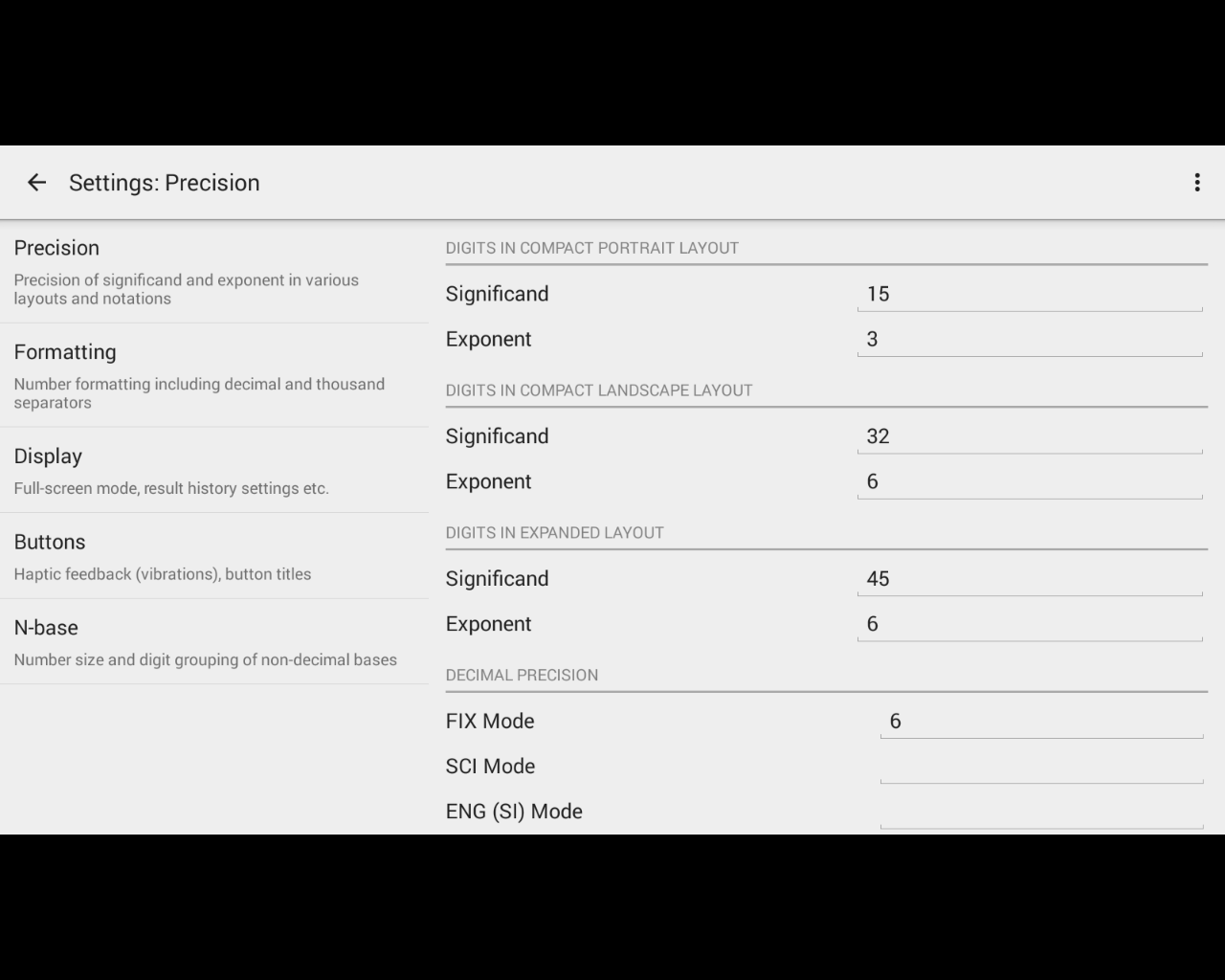
I will attempt to include this feature in my own project because it allows a larger percentage of users to access the program and therefore become more useful to a larger number of people.

Another helpful feature that this app includes are the tips that come up when the user accesses certain, more nice or complicated features. 

*A tip introducing the user to converting between different displays*

This feature makes the calculator more accessible to users with different levels of knowledge and experience with using scientific calculators. This will be especially important in my own project because increasing accessibility to a higher proportion of users means that more students will be able to benefit from the app. Therefore, this will certainly be a feature that I will be implementing into my own project. I will focus the use of these tips on underutilised features such as the memory storage because it’s important not to bombard the user with notifications as it can annoy the user and so make the app undesirable.

This app also has a very large range of settings for different parts of the program, such as formatting and the display.



*The different settings for ‘precision’.*

The range of settings for the calculator provide a good benefit for using the HiPER Calculator over other apps because it offers a large amount of customisability, making the app easier for the user to interact with and understand.

This app has other features designed to heighten its level of understandability such as the larger range of error messages for different situations.



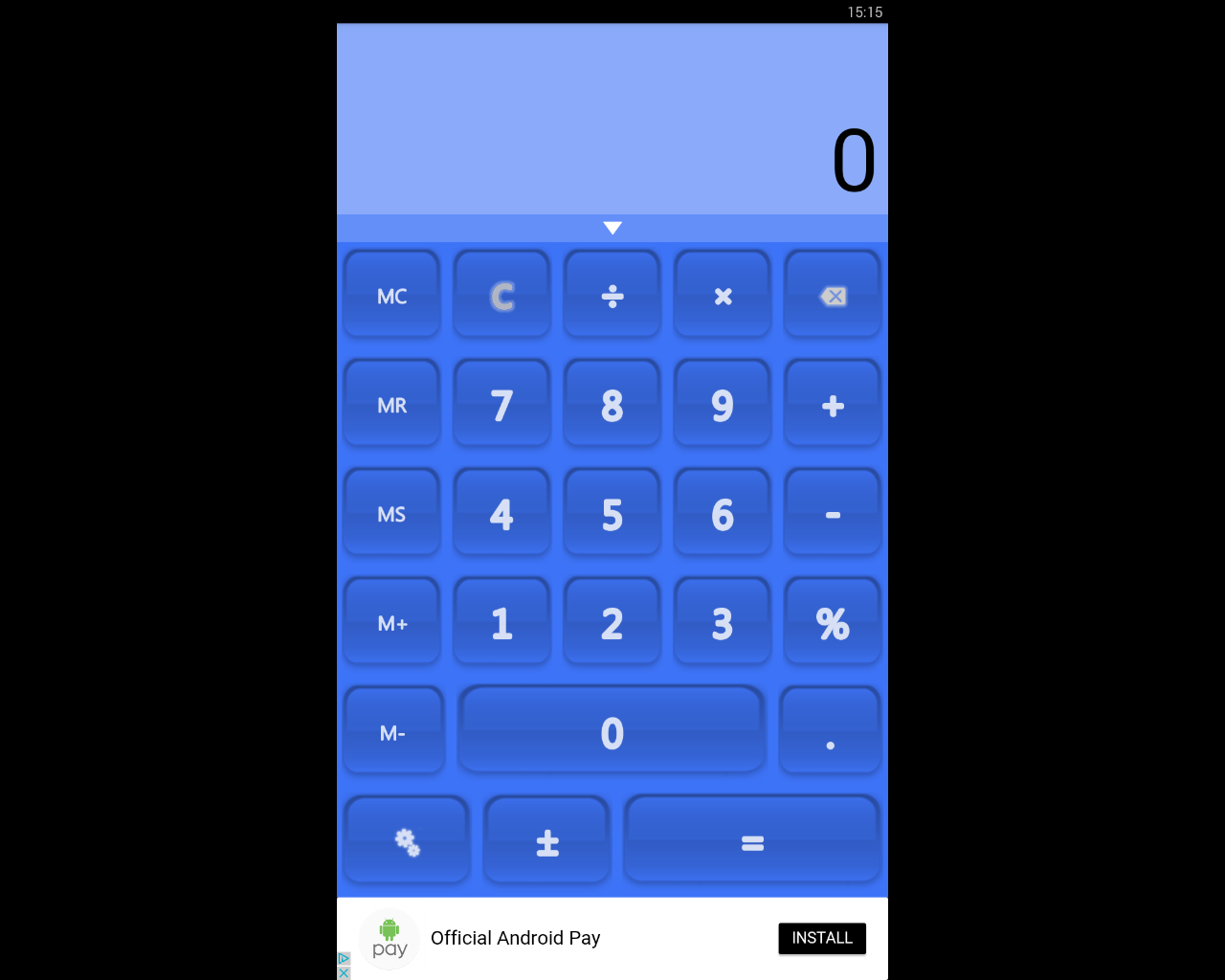
*The error that appears when an expression is unfinished*

This feature is designed to help the user and make the functionality of the app easier to use and understand because the error messages are more descriptive than tradition calculators which would simply output ‘SYNTAX ERROR’. The traditional calculators are less detailed therefore require the user to take longer to fix their expression, whereas the HiPER Calculator has a detailed error message and even highlights the area where the error occurred.

This, and features similar to it such as the calculator auto-completing the expressions in some situations such as a missing bracket or unused decimal point improve the calculators usability by streamlining the process for the user.

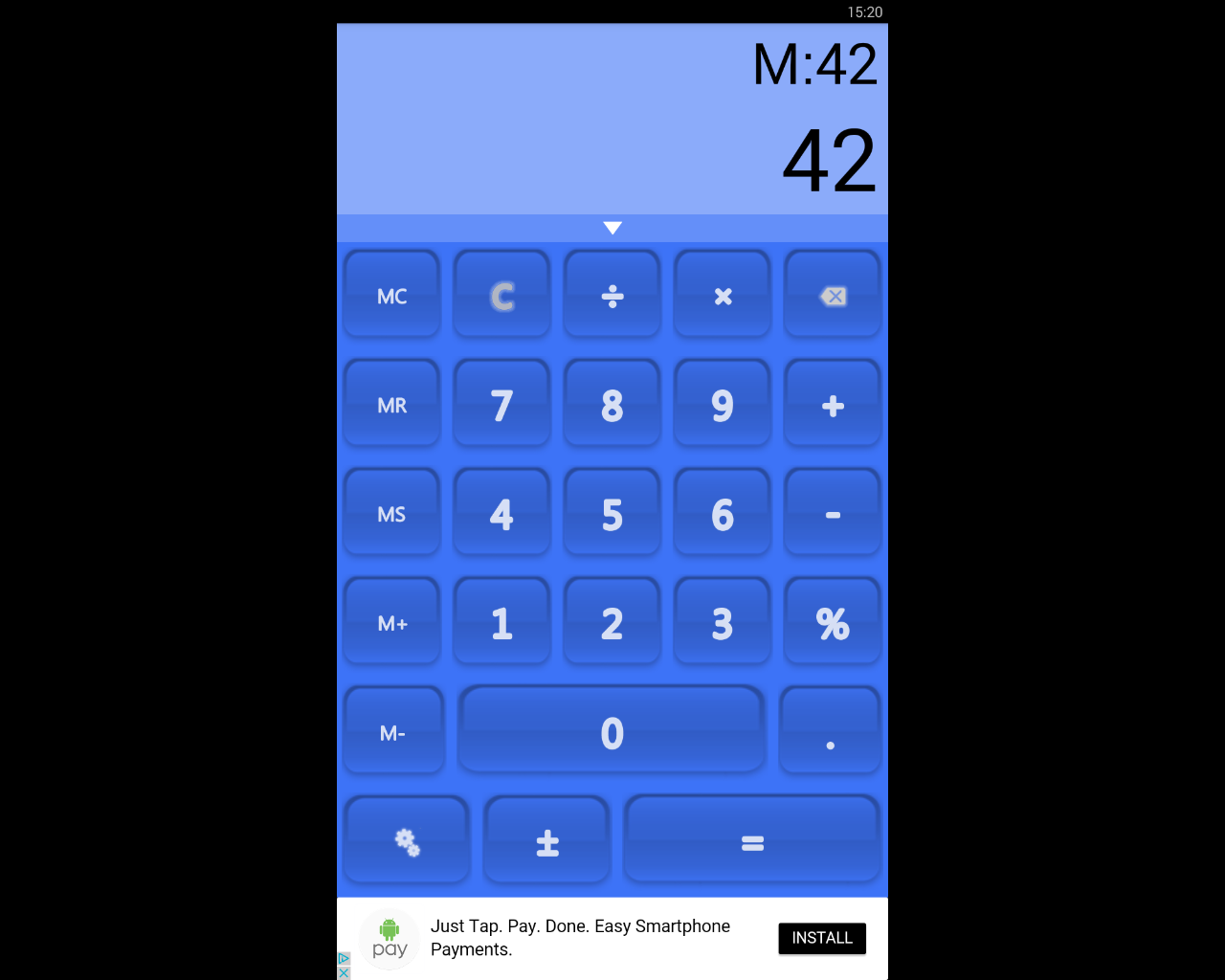
### 1.3.2 ColorFul Calculator

This app is an example of a less successful, while still functional calculator. The presentation of this calculator is lacking, especially in terms of a professional setting.



*The base view of the calculator*

While this app does cover all of the very basic functionality, such as the four basic operations, capability for percentages, and memory storage. However, the app does not extend much beyond that. For example, although the app does contain the ability to store values in the memory, there is only space for a single number.



*The value ‘42’ is stored in the memory*

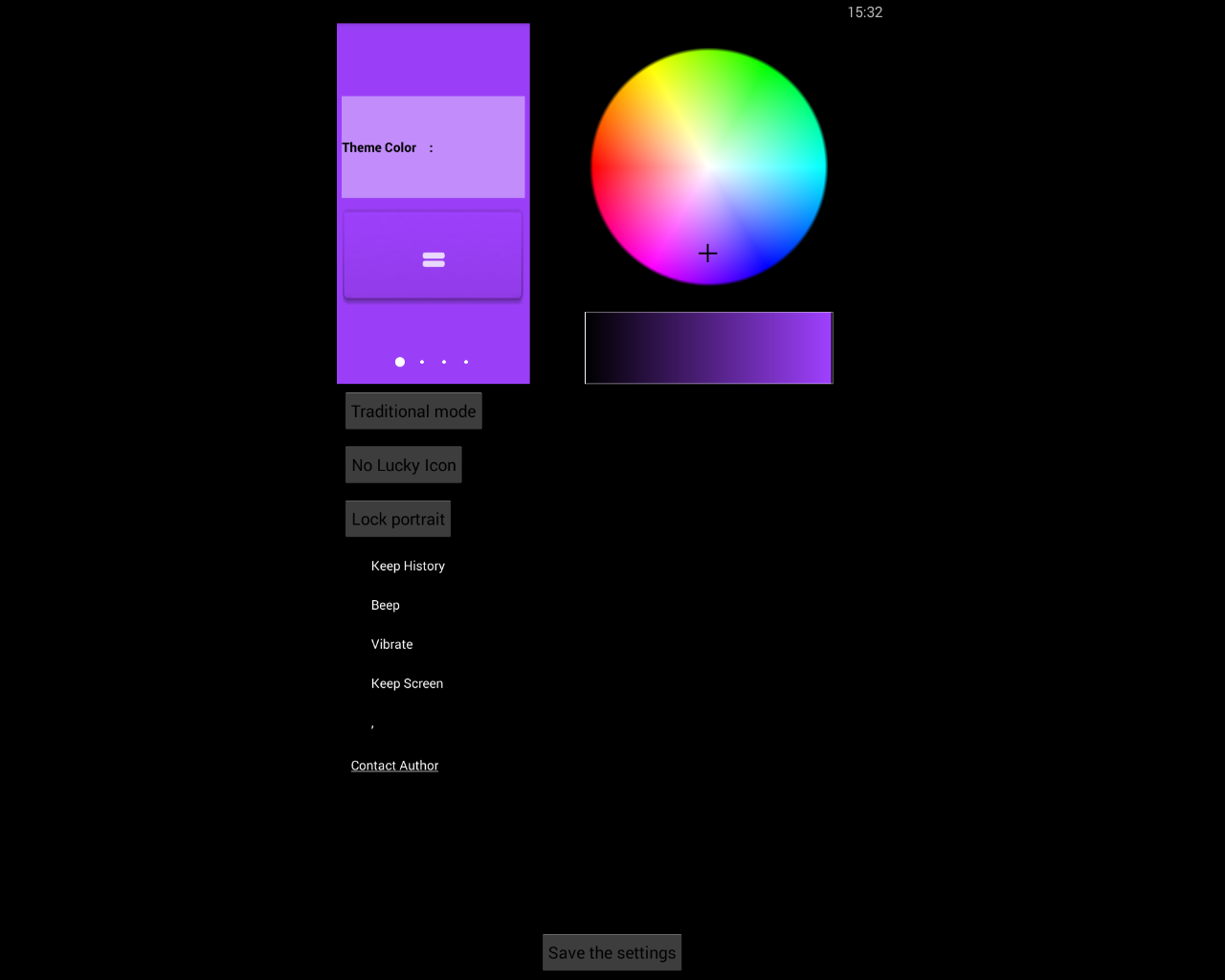
Due to this lack of any major functionality, the app is much less successful when it comes to the user performing calculations that are more involved than operations on a couple of numbers. Another disadvantage to this app is that when inputting the expression, the calculator does not display the user’s input. This means, if the expression is longer than a few separate numbers, it can be awkward to remember what the user has entered to the calculator, meaning it is easy to make mistakes.

Another disadvantage due to this flaw is that if the user gets an incorrect output, they cannot de-bug their own expression to find where the error occurred because it is not displayed to them.

Furthermore, the app contains advertisements which can be very intrusive based on their location. As shown in the screenshots above, the adverts are placed below the calculator, just below the ‘equals’ sign. This placement could lead to the user accidentally clicking on the advert instead of the equals sign if using the app quickly. This could be very frustrating for the user, leading to less usage of the calculator.

In light of this, I will not include adverts in my own project because I think they can be very detrimental to the user’s experience. Also, if it was made necessary to have adverts, due to a need for funding, for example, I would make sure that they were in the most unobtrusive location possible because the user’s experience should be the utmost important.

However, this app does have some redeeming features, for example, it has some semblance of customisability.



*The range of different colours that can be selected*

The app can be customised using a colour wheel that changes the general theme of the rest of the app. Although this feature does add customisability to the app which is a positive, the potential customisation is largely unprofessional and gaudy making it ill-suited for an app that will be used in a work environment. Therefore, in my opinion, this feature would be largely pointless in my project.

This app does contain other advantages, such as the feature that includes a history of past calculations.



*The past calculations are shown*

This feature can be very useful because it allows the user to access and help remember what their past calculations are, and so reduces the chance of the user losing track of what they’re doing.

## Requirements for the Solution

Requirements for my solution can be split into two sections, the requirements for myself as the developer and for the user.

### 1.4.1 Developer’s Requirements

The resources required for the solution to be developed will include things like:

* A computer with the capability to run Java. This is because I will be programming the project in Java due to the language’s versatility and because I am very familiar with it. This computer will need to have no limitations or restrictions on the internet usage and will require full admin access.
* An Android virtual machine. This will be used to test my program without having to use a physical phone which will make the process quicker and more efficient.
* An Android phone with the required minimum specifications. This will be used for testing of the project, so I can find out how the program runs on actual phones and use it to bug fix my program.
* Testers. I will require GCSE students to test my program on because they are the primary stakeholder and, so it will be important to test my program on them to take their criticisms into account to make the program as useful and successful as possible.
* Access to wherever the app is hosted, for example the Google Play Store so the app can be accessed easily by the users.

### 1.4.2 User’s Requirements

The user’s requirements will differ greatly from the developer’s. Their requirements will contain:

* An Android phone with the minimum specifications requirements, including the correct version of Android. This will mean that they will be capable of running the program without crashing or otherwise harming their phone. Also, their phone will need to have the correct resolution so that the program can be displayed clearly, and the UI is properly presented and is easily used and intuitive.
* A stable internet connection. This will be required so that the user can download the app from wherever it is hosted and also to potentially download updates.

## Limitations of the Proposed Solution

My proposed solution will, unfortunately, have some limitations. For example, my project will likely have the limitation of not having functionality for graphs. While having the ability of drawing graphs in the app would be very useful, I think it will be very difficult to be able to include this feature.

This is because implementing the feature will have more difficulties to overcome than other, more simple features. Also, graph functionality is less of a priority compared to other features that are simply more important.

This can be seen in that there are dedicated apps to just graphs, showing that it is very complex to make well. Because of this, I think that if I try to implement graph functionality, other parts of my project will suffer.

Another limitation that is likely to occur is that the app itself may end up with an ugly user interface. This is because I am lacking when it comes to making artful and attractive design. Therefore, to overcome this potential limitation, I will certainly make often checks and tests with other people such as potential users. This will ensure that the final product will have an attractive design that is easy to use and intuitive.

## Success Criteria

The success criteria for my project can be broken down into basic features that are critical to the apps success and more advanced ones that, while still important, the app can function without.

### 1.6.1 Basic Success Criteria

* Functionality for the four basic operators, +, -, x, ÷. This will be of the utmost importance because without this, the calculator will not work at even the most basic level.
* The calculator will follow the rules of BODMASS. This feature is very important because it makes the experience more streamlined for the user because they don’t have to concern themselves about the formatting. Also, it will allow users to be more familiar with the app as they will be used to calculators following BODMASS.
* Have functionality for using square roots and powers. This feature will increase the amount of calculations the user is able to perform.
* Be able to use sin, cos, and tan and the inverses of. This is feature is required because it will enable the user to do trigonometry related functions which are a very large part of the maths GCSE.
* Cursor controls that will move the cursor around the equation. This is required because it allows the user more freedom and enables the user to edit equations.
* Capability for common forms of notation, such as standard form, fractions, decimals, etc. This is essential to the app because it will allow the calculator to be more understandable for the user.
* I will use a survey that asks people that have used the application to rate the usability and intuitiveness of the GUI. If at least 80% of the answers give positive feedback, I will consider this success criteria completed. I can also use this feedback to improve upon the design in later development iterations.

### 1.6.2 Advanced Success Criteria and Essential Features

* Displaying tips when the user access more niche and unknown features. This feature will be very beneficial because it will help users who are less computer literate and therefore increases the accessibility to a higher proportion of the potential user base.
* Store equations for subjects such as Maths, Physics, Chemistry, etc. that the user can select and enter their variables to complete the equations. These equations will be stored in a list that the user can access by tapping a button on the main activity.
* Detailed, helpful error messages that inform the user. This is important because it enables the user to use the app in a more productive way and therefore improves the apps usefulness.
* The capability to change the layout of the app to accommodate users with less common device sizes. This feature is important because it will allow a larger percentage of users to be able to interact properly with the app.
* Conversions between different base number systems, namely, binary, hexadecimal, and denary. This will be a useful feature as conversions between number systems feature heavily in the Computer Science GCSE and therefore will be helpful to GCSE students.
* Conversions between different units, such as miles to kilometres. This feature will be very beneficial to the app because it will streamline the user’s process when doing calculations because less time will be needed to be spent doing conversions.

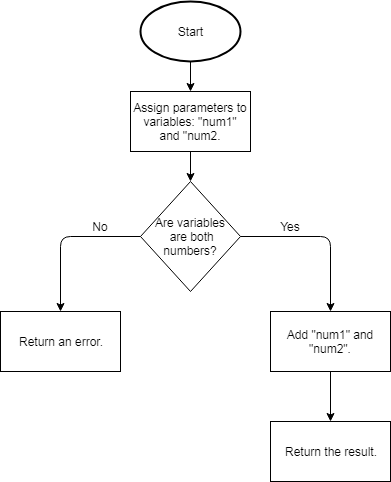
# Design

## Method Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Method** | **Parameters** | **Variables** | **Class** | **Description** |
| BODMAS | The mathematical expression that the user enters that will be calculated. This is called the infix. | * infix * postfix | ShuntingYard | This method is the basis of the entire app. This will take a string that the user provides, called the “infix”.  I will use the Shunting Yard algorithm to apply BODMAS. This algorithm is stack based and works by taking the infix (which is the user’s mathematical expression) and converts it into the postfix. The postfix is created by manipulating the infix with stacks. The postfix is also known as Reverse Polish Notation or RPN.  The postfix can then be evaluated to compute an answer to the user’s original expression. This will occur in a different method.    For example, if the infix is “3 + 4 × 2 ÷ ( 1 − 5 ) ^ 2 ^ 3” the postfix will become “3 4 2 × 1 5 − 2 3 ^ ^ ÷ +”. |
| EvaluateRPN | The postfix that has been created by the BODMAS function. | * postfix * result | ShuntingYard | This method will evaluate the postfix and return the answer to the user’s original expression.  This is done by looping through the string of the expression and when the specific element is an operator, pop two numbers off the stack and manipulate them according to the type of operator that is given.  For example, if a “+” operator is the current element in the loop, two numbers will be popped off the stack and added together. This works similarly for other basic operations such as powers, division, subtraction, etc.  Because the string is in Reverse Polish Notation, the order of the numbers and operators means that when the numbers are popped off the stack, they will be calculated according to BODMAS. |
| Roots | A float as the base and an integer as the root. | * base * root * result | MainActivity | The user can enter their root number and their base number. The base number will be rooted by the root using the Java Math class.  The result will then be stored in the result variable.  For example, the user entering “2” and “4” will perform a square root of 4. |
| Trigonometry | Which function is to be used (sin/cos/tan).  The number that it should be performed on. | * trigFunction * input * result | MainActivity | The method will evaluate the trigFunction variable to decide the trigonometry function that will be used.  Depending on the trigonometry function, the user’s input will be calculated using Java’s Math class. The answer from this will be stored in the result variable. |
| Inverse Trigonometry  (sin-1, cos-1, tan-1) | Which trigonometric function is to be used (sin-1/ cos-1/tan-1).  The number that it should be performed on. | * trigFunction * input * result | MainActivity | The method will decide which inverse trigonometric function to use on the user’s input depending on the parameters given to the method.  When the calculation is done, the result will be stored in the result variable.  This method will require validation so the user’s input are within the appropriate bounds. For example, doing the calculation sin-1(x) where -1 ≤ x ≤ 1 will result in an error. Therefore, the method will require validation to prevent the user performing these actions and crashing the app. |
| convertNotation | Which notation form the user wants to convert their answer to.  The notation that the user’s number is currently in.  The number that the conversion will be performed on. | * currentNotation * requiredNotation * input * result | MainActivity | This method will take the user’s input and convert it to all the other notation forms. This will be displayed to the user.  For example, if the user has a decimal number, it will be shown in fractions, standard form, and surds.  The user then has the option to select a specific type of notation for their input to displayed in. |
| convertUnits | A string containing the current unit and a double containing the value of that unit. | * currentUnit * input * resultNum * resultUnit | MainActivity | This method will be called on the user’s click of a button. It will bring up a menu where the user can select the conversion they want. For example, they can select “cm -> m”.  The method will either take the number currently stored in the calculator’s answer variable or the user can enter their own.  In this example, where the user has just entered a calculation with the result of “120.0”, the currentUnit would be “cm”. The input is “120.0”, the resultUnit is “m”.  The calculator will then perform the conversion of centimetres to metres and store the result in the variable “resultNum”.  In this example, the resultNum would be “1.2”.  The method will have the functionality to convert:   * distance * time * area * mass * volume * speed |
| Display Error | The error code | * errorCode * errorDetail | MainActivity | This method will be called for different types of errors.  For example, it will be called when the calculator encounters a failed verification check. When this occurs, the error code will be a parameter in the method call.  This error code will be linked to a description of the error and how to fix it. This description will be a string stored in the variable “errorDetail” and will be displayed to the user.  Another example of when this method might be called is when a method catches an exception. Each type of exception will have its own unique error code with a corresponding description of the error that will be displayed to the user. |
| showEquations |  | * selectedEquation | Equations | This method will display a list of the stored mathematical equations in the calculator.  When the user selects one, this method will call “useEquation(x)” where x is the user’s equation. |
| useEquation | The equation that the user has selected | * Variables that are used in the equation, such as frequency, distance, speed, time, etc.   These will be provided by the user.   * result | Equations | Upon being called, this method will query the user for to provide the known variables and select the unknown variable.  For example, with the equation to calculate speed/time/distance, the user would provide two variables – “30 metres” and “5 seconds”. Then the user would select solve for speed and the calculator would output “6 metres per second”.  All equations require the values to be entered in SI units. This is because the equations in GCSE and A Level papers are to be used with SI units. Therefore, this will help the user become used to doing calculations and become better prepared for their exams. |
| changeLayout | The new layout that the user has selected. | * currentLayout * tablet * phone * portrait * landscape | MainActivity | This method will change the xml layout of the GUI. This will accommodate different screen sizes, so all users can use the app easily without elements being cut off or inaccessible.  This method will also enable the users to switch between portrait and landscape layouts. The different orientations will require separate xml files because each one will need to be designed for the all the features to be intuitively accessible. |
| showTip | A variable that identifies which function the tip should be shown for. | * tipDetail | MainActivity | When a method is accessed for the first time, it will call this method.  This method will display a pop up to the user that explains the method that the user is accessing does and anything else that will help the user to use the app. The content of this will be stored in the variable “tipDetail”.  For example, when the user first accesses the convertNotation() method, a pop up will be shown explaining that the method shows different notations for the user’s results. It will also outline how the user can select a notation to be used in the calculator. |
| moveCursor | The directional input (this will be decided depending if the user clicks the left or right arrow). | direction | MainActivity | Moves the cursor in the GUI different direction depending on the user’s input. |

## 2.2 Algorithms

### 2.2.1 Basic Operations, e.g. Addition

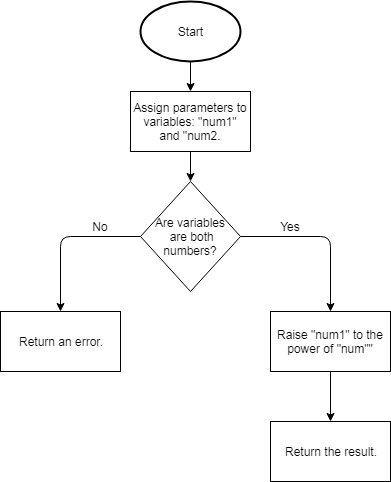


This flowchart shows that two variables are created and added together, then the method is returned.

A similar algorithm will be used for the other basic mathematical operations such as subtraction, division, and multiplication. This is done by changing the operation that is done between the two numbers.

This method is required because basic mathematical operations are the minimum requirement for a calculator to be of any use. Even if the calculator had more advanced features such as converting numbers to different number bases, the calculator would still require basic functionality for it to be helpful to any user.

### 2.2.2 Powers

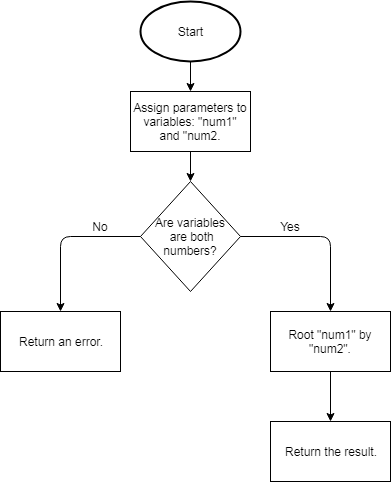


This flowchart shows how the initial parameters that are passed to the method are assigned to the variables, “base” and “power”. After validation checks are performed on the two input numbers, the calculator will raise the base to the order of the power.

This method is required for the calculator app because the ability for the user to user more advanced maths (compared to basic functionality) is very important to for students in their GCSEs or A Levels.

Powers are used in almost every aspect of a STEM subject in GCSE and therefore this functionality is required for the calculator to be of use in more situations.

### 2.2.3 Roots

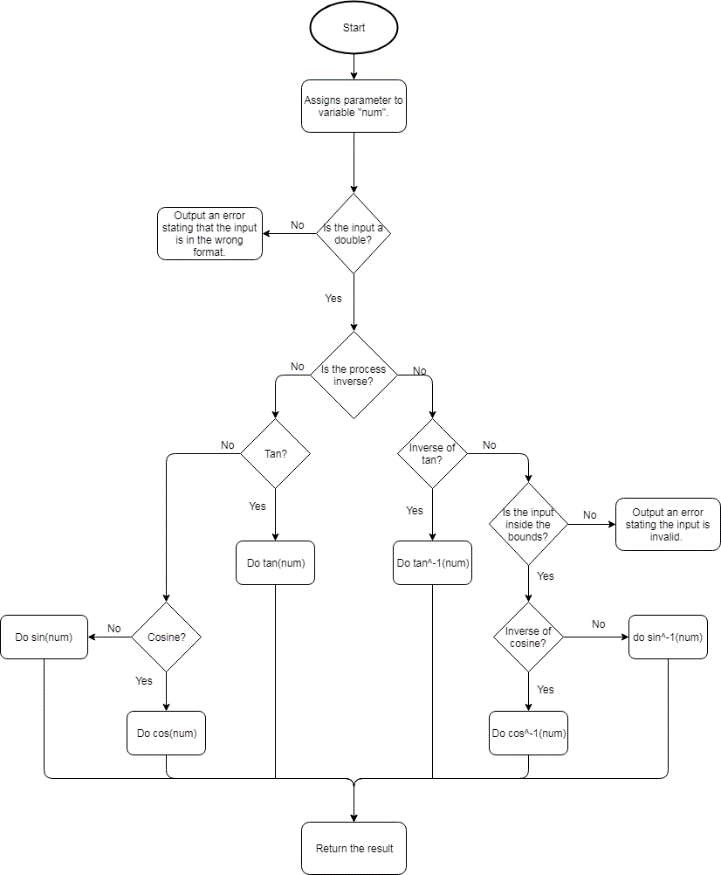


This method takes the parameter and assigns them to the variables ‘base’ and ‘root’. The two variables under go a validation check to ensure that using them won’t result in an error that could lead to a crash.

After the validation is confirmed, the the “base “variable will be rooted by the “root” variable.

This method is required so the calculator can be used and be more helpful in more situations. This is because the functionality to root numbers is often needed during school lessons and for schoolwork. Therefore, without this feature, the calculator would be useless in many situations.

### 2.2.4 Trigonometry



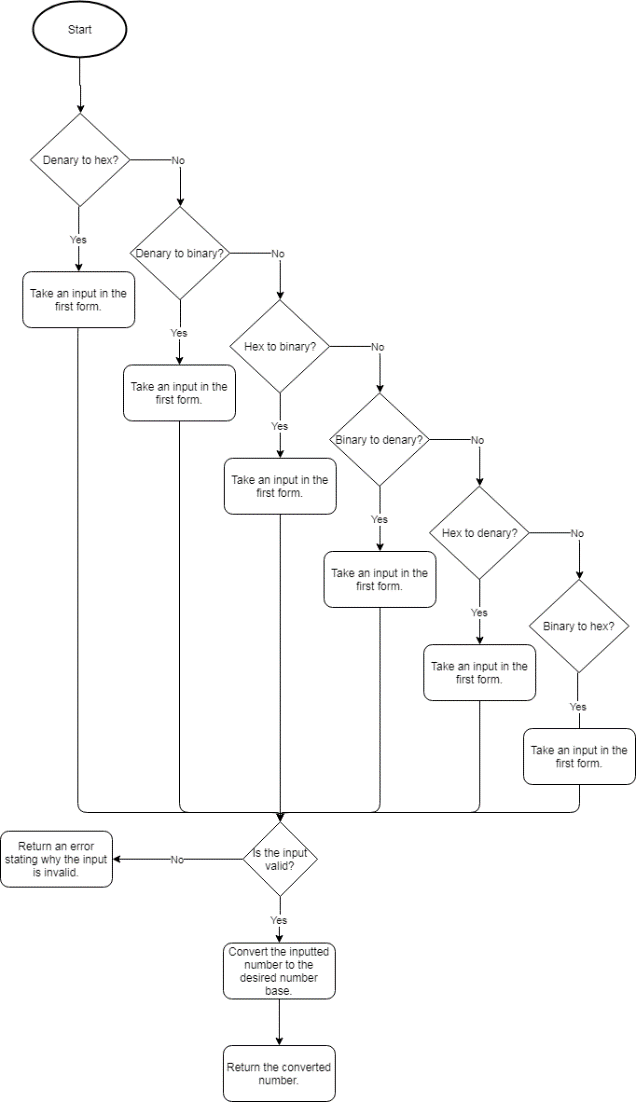
This flowchart shows how the function for calculating trigonometric expressions will be carried out.

This includes two forms of validation. The intial validation ensures that the input is of the correct data type so the function does not attempt to perform a calculation on a string.

The second form of validation is for the sine and cosine calculations only. This validation ensures that the input is between 0 and 1 for inverse calculations of sine and cosine. The validation is only required for the inverse versions of sine and cosine as they must be within the maximum and minimum bounds of their waves whereas the tan wave increases to infinity.

This method is required because trigonometric functions are often used in GCSE and A Level subjects. For example, trigonometry is a large of the Maths GCSE specification and every student will need to learn how to use it. Having this feature in the calculator will help the user understand trigonometry and so help them in their exams.

### 2.2.5 Number Base Conversions



This flowchart shows how the process of the user selecting a type of conversion, then entering a number which then gets converted.

This algorithm enables the user to perform conversions to different bases such as binary to hex.

There is one instance of validation in this algorithm. After the user inputs their number, it will be run through the validity check. This checks that the input is valid for the different types of conversion, for example, binary to denary will only accept the numbers ‘1’ and ‘0’.

This algorithm is especially required for Computer Science GCSE and A Level students. This is because the Computer Science specification requires students to convert denary to hex, binary to denary etc. The specification also includes basic binary operations.

Therefore this feature will be very helpful to those students as it will allow them to check their working so they can see their mistakes and improve upon them.

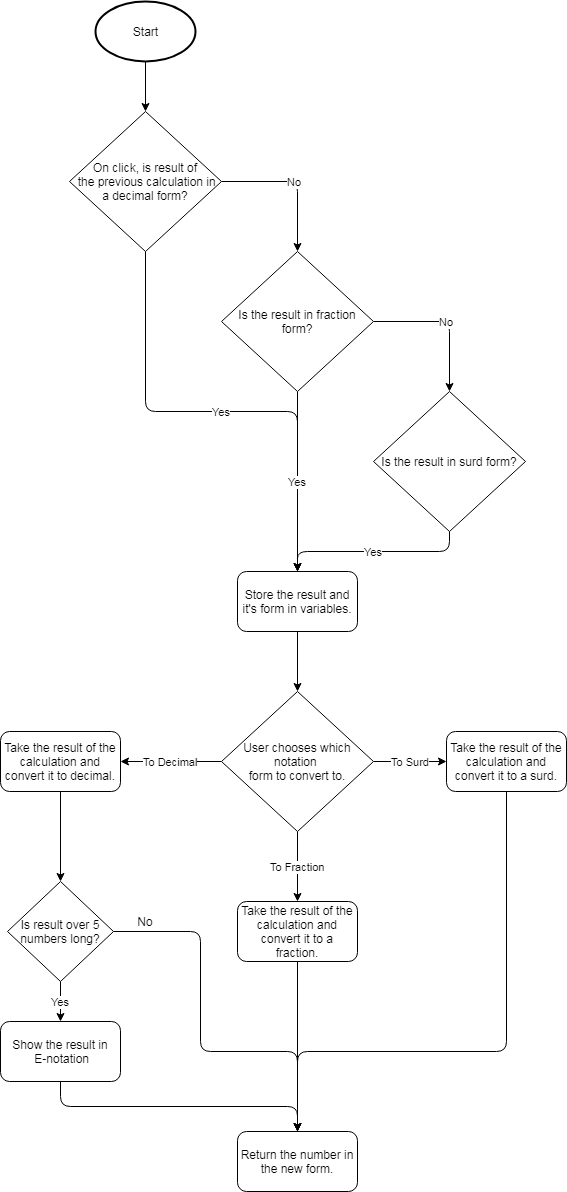
### 2.2.6 Number Base Conversions Validity CheckC:\Users\2dennyd\Downloads\Number Conversion Validity Check.png

This flowchart shows how the input for the base number conversion will be validated. The algorithm will employ different validation methods depending on the resulting number base that the user wants.

For example, if user selects binary to denary conversion, then the input number must only contain the digits “1” and “0”. This is because binary is a base 2 number system.

This function works in conjunction with the number base conversions. Therefore, this algorithm is required so that the number base conversions can be used without any errors occurring. This potentially prevents errors that could crash the app or otherwise influence the user’s experience.

### 2.2.7 Notation Conversion



This flowchart shows the process of converting values between different types of notation.

For example, if the user chooses to convert a fraction to a decimal, the first decision (is the input decimal) will be a no so the flowchart carries on to the next one. This one is positive because the decision is whether the input is fractional. The input and which notational form it’s in is stored in variables. Then the user chooses what notation to convert it to.

In this example, the user chooses decimal notation. The input is converted to decimal and another decision occurs, if the decimal result is over 9 digits long, it is converted and returned in scientific notation (used when the decimal number is too big to be conveniently written).

This method is required because throughout school, the students are required to be able to convert fractions to decimals and vice versa and to be able to use surds, etc. Therefore, having a method which will display the other forms of their calculations will help the students become familiar with all the different forms of notation.

### 2.2.8 Unit ConversionC:\Users\2dennyd\Downloads\Convert Units.png

This flowchart shows how the method for converting units will be carried out.

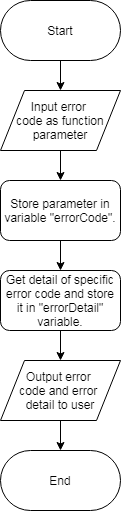
For example, the user will choose their conversion such as millilitres to litres. The method will check if there is currently a number stored in the answer variable. If so, the method will display the conversion using the answer variable as the input number.

After displaying it (or straight away if the variable is empty), the method will then ask the user for input.

If this is denied, the method will exit and the user will be returned to the main calculator screen.

This method is required because in STEM subjects, especially ones like Physics, equations are often used with different units. The specification requires the student to understand how to convert units and so this calculator will help them understand the process and let them focus on harder parts of the problem.

### 2.2.9 Display Error



This flowchart shows the process of an error occurring and it being represented to the user.

For example, this algorithm will be a method that is called when an error occurs. The method call will have a parameter which will be an integer error code.

Each unique error code will have a corresponding string that can be displayed to the user. This will explain the reason of the error and how the user can fix it.

This algorithm is required because having a clear and understandable error messages will be very important when it comes to the apps usability and the user’s experience.

For example, the user makes a mistake when they write out their expression and it doesn’t work. If the app doesn’t have any intuitive error messages the user will become frustrated and annoyed. This ruins the user’s experience and makes it less likely that the user will continue to use my app.

Therefore, having clear, concise and helpful error messages is an important requirement of a successful solution because it greatly improves the UX and makes it more likely that the app will be used.

## 2.3 Usability Features

### 2.3.1 Learnability

To help the users learn how to interact with the app quicker, hints will be shown when the user clicks on different buttons. For example, when the user enters an equation that outputs a surd, the app will notify the user of the feature that converts values from surds to fractions or decimals. It also shows the user how to use the feature.

Another way the app will be made easy to learn is through the design of the user interface. The UI will take design features from other well-known calculator apps and real life calculators. This will increase the learnability because the user will find the design intuitive to use.

For example, it will use a “shift” button. This is a commonly used feature on calculators that enable buttons to double up on uses. The buttons can have a default function, and a shift function. Using this makes the app easy to learn because the “shift” button is often used in calculators and so it will be intuitive and not require much time for the user to learn.

### 2.3.2 Efficiency

To help the users interact with the app as efficiently as possible, the majority of the actions the user can take are located in the first activity and layout of the app. Therefore for a large percent of the time that the app is being used, all the usage will remain in the first activity and therefore it will be efficient. It will be efficient because the user will not have to trawl through menus to perform their desired action.

Another way the efficiency is improved is by multiple layout options for different device dimensions. For example, a tablet layout will take advantage of the larger amount of space to include more buttons and functionality into the main layout. This means the space on the screen will be used very efficiently to enable the user to perform functions as quick and efficiently as possible

### 2.3.3 Memorability

Using the app will be very memorable because it will closely mirror the design of real life scientific calculators. This means that most features will be labelled and have the same annotations as a real calculator. This will help with memorability because the userbase will likely already have experience with scientific calculators. Therefore they will be used to interacting with a similar design and so they will remember how to use it.

### 2.3.4 Errors

My app’s layout and functionality will be designed to maximise the ease of use for the user. The most common error would likely be the ones caused by the user entering invalid data that could cause crashes. To prevent these types of errors, I will include a lot of validation for every method that the calculator uses. For example, when the trigonometry method is used, the input will be checked that it is between -1 and 1 for cosine and sine.

If the input is invalid, using it in the method would cause an error that could cause the application to crash or otherwise stop working. Instead, the validation rules can catch it and instead inform the user of the invalid input and prevent it from being used.

Other common errors that the application will encounter will be calculations have a result that is outside the possible range for the application to process. This will occur when the user’s expression results in an output that is too large to either display on the screen or the processor cannot handle the process.

The error of having a result that is too long can be solved by detecting when the error occurs by testing the length of the result and then changing the notation of the result if it’s too long. For example, converting the result from decimal to scientific notation will conserve space and enable the user to read the result with more ease.

The error of the user entering an expression that uses all of the CPU’s processing power which will cause the device to freeze or crash is more likely to occur on older devices, such as the one’s that run on Android 4.0 and older. This error will be much more difficult to prevent because different devices can handle different levels of processing power, therefore the application cannot be programmed to reject specific equations that are known to crash a device.

### 2.3.5 Satisfaction

My app will be as satisfying as possible for the user to interact with. I am aiming to make it satisfying by creating the design based around mainstream calculator designs that are already widely used.

For example, I can base my calculator’s layout off the common calculator that students and schools use. This will make my app more satisfying to use because the user will have experience with other calculators that can be transferred over to my own.

Because of this design’s easy accessibility, the potential user base will be as wide as possible. Therefore the solution can be used by any student, teacher, or parent no matter their age or level of computer literacy. This means each user will be satisfied with the solution because the design will be specifically created so it is easily accessible to sany user.

I will also carry out a survey targeted at GCSE and A Level students to ensure my design is aesthetically pleasing and can be easily interacted with. This will ensure that my final design is as satisfying as possible.

### 2.3.6 Design Drawings



## 2.4 Validation

### 2.4.1 Validation Features

Many of the features that will be included in my project require very specific inputs. Therefore, I will have to implement many validation rules for each feature.

For example, I have many features that require an input between specific ranges. An input that is outside the range could cause an error which could cause the program to have an unrecoverable crash. Therefore, I will include code that validates the user’s input on a case by case basis when each feature is used.

This validation will include things such as checking that the user’s input is within the correct range. For example, with the inverse sine and cosine functions, the input must be within -1 and 1. If the input is outside of this range, a message will be shown to the user detailing the message and how to fix it.

The message to the user will be shown instead of running the function with the invalid input. Therefore, the crash will be prevented and the user can fix their error.

Another potential error is from the user inputting an incorrect expression. For example, having multiple operators that don’t apply to a variable. E.g.: “3+\*2”. This will cause an error because the “+” operator is trying to perform an operation on “3” and “\*”. The data types are incompatible and therefore it will cause an error.

This can be prevented using validation rules that will not allow operators adjacent to each other in the expression. However, exceptions must be made to the minus operator to allow for negative numbers to be used.

When this error is caught by the validation rules, a message will be shown to the user that explains the error. Also, the cursor will be placed at the point where the error occurred to make it easier for the user to find their mistake and correct it.

## 2.5 Testing

### 2.5.1 Iterative Test Plan

To ensure that my program is working properly I will use a range of data to verify that the program interprets the data correctly and processes it without any errors.

To ensure that the tests are carried out properly I will need to use data that mimics every possible input the user can access, such as a range of numbers and characters.

This is required because I need to test that the program correctly processes the data that is supplied to it. All the possible inputs are needed so that a bug doesn’t go unnoticed through the iterative development process. This could cause much larger problems further along the development cycle.

I will be testing my application through the use of unit tests. Each function that I implement will have a range of unit tests ensuring that it works correctly. The unit tests will take a variety of inputs that cover any possible scenario that the user can do. Therefore, the unit tests will ensure that the application responds properly to every possible input for every function.

An example unit test for the addition / subtraction function:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Input | Reason | Expected | Actual | Pass/Fail and Actions |
| 1 + 2 | Adds two positive numbers | 3 |  |  |
| 1 + + 2 | Two addition operators should result in an error. | Error Message |  |  |
| 1 + - 2 | Adds/subtracts a negative and positive number. | -1 |  |  |
| -1 - 2 | Subtracting 2 negative numbers | -3 |  |  |
| -1 + - 2 | Subtracting 2 negative numbers with an addition operator | -3 |  |  |
| 1 - - 2 | Two subtraction operators should result in addition. | 3 |  |  |

### 2.5.2 Testing of Final Implementation

The final implementation will be tested with a white box method. This will be done by myself as I will be most knowledgeable about the program and so I will be most qualified to test every aspect of the program.

This is an important testing method to use because it will ensure that every function of the programs works as intended. This may not happen with black box testing because the people testing the program are not familiar with it and therefore they could miss errors.

Therefore, white box testing is very important as the developer is most likely to recognise every error that occurs so the final implementation will be as bug free as possible.

I will be testing the final implementation’s user interface using black box testing. I will create a survey that has brief instructions of how to use the program and ask people who have little knowledge of the system to follow the survey’s instructions to use the program.

I can use the survey to track how easily people can interact with my program with minimal guidance. This should test how accessible and user friendly my final implementation is. The use of a survey will also make it easy to get feedback on the user interface to make any potential improvements.

The use of black box testing will provide an unbiased perspective on the app. This will generate valuable feedback that I would not otherwise be able to get using other testing methods.

The final method that will be used is destructive testing. In this testing method, the developer tries to break the system when in full use.

It’s important to use this method of testing with the final implementation because the final product should not have any fatal bugs that lead to a full crash. Destructive testing will reveal any of these potential fatal errors so safe guards can be put in place.

This type of testing will therefore ensure that user is provided with the cleanest possible experience and never experiences any fatal errors that crashes the application and interrupts the user.

In conclusion, I will use a combination of white, black and destructive testing because this will provide the largest range of different tests which should cover any potential errors that will affect the user’s experience.