# Software Requirements Specification

CALCULATOR PLUS

Group 8: Gabriel Magallanes, Mohammad Mirwais, Sean Mitchell
CALIFORNIA STATE UNIVERSITY, FULLERTON | CPSC 362

## Contents

## Introduction

### Purpose

- This program aims to create a fun and interactive way for kids to learn basic math. The application should be easy to use, while also being fun to interact with. The application will generate random easy math problems limited to addition and subtraction and prompt the user to enter in an answer for the randomly generated problem. The application will provide feedback for the user and tell them if the answer was correct or incorrect. The user will also have a limited number of attempts to input the correct answer. Again, the main underlying goal is to provide a fun and simple way to teach a younger audience basic math.

### Intended Audience

- The main target audience of this application is children, specifically first graders. Due to the younger audience, the application must be simple and easy to use. The parents of the first graders could also be a possible target audience, so the application must be family friendly and easy to use for the parents as well.

## Code Deep Dive

The intended purpose of this section to provide documentation on how the code works. Function arguments, descriptions, outputs, and inputs will be described in more details here.

`class Application(Toplevel):`

- The main class for the application which holds all the application variables and functions.
- Variables:
  - `image` – source file image to represent adding/counting objects.
  - `image2` – another variant source file image to represent adding/counting objects.
  - `bgimg` – the background image for the application window.
  - `width` – stores the width of any of the source file images.
  - `height` – stores the height of any of the source file images.
  - `thisEntry` – Entry widget for the user to type in any input.
  - `maxlabel` – warns the user that they have used up all their guess attempts.
  - `nextButton` – Button used to navigate to the next question.
  - `exitButton` – Button used to exit the application.
  - `addlabelx` – stores the width of any of the source file images.
  - `addlabely` – stores the width of any of the source file images.

`def calc_For_Kids(self):`

- This is the main math game function. This function controls the game loop and any events that will happen during the game such as input and validation. Will make use of the checkAdd, addFunc, checkSub, subFunc and arrange_ball_images functions.

`def checkAdd(self, a, b):`

- This function checks if the user entered in the correct answer.
- Arguments:

  - Self:
  - a: first number input
  - b: second number input
- Output: Boolean value whether the user entered in the correct answer.

`def addFunc(self):`

- This function adds two numbers together
- Arguments
  - Self
- Output: does not return any value, this function is used to add two random numbers.

`def checkSub(self, a, b):`

- This function checks if the user entered in the correct answer.
- Arguments:
  - Self:
  - a: first number input
  - b: second number input

Output: Boolean value whether the user entered in the correct answer.

`def subFunc(self):`

- This function subtracts two numbers.
- Arguments
  - Self
- Output: does not return any value, this function is used to subtract two random numbers.

`def arrange_ball_images(self, sign, a, b):`

- This function will generate an array of ball images on the user's screen to visualize the randomly generated math problem.
- Arguments:
  - Self:
  - Sign – the operator used in the randomly generated math question
  - a: the first number input in the randomly generated math question.
  - b: the second number input in the randomly generated math question.
- Output: displays an array of ball images, no return value.

## Project Requirements

### Splash Screen

- On program startup, the user will see a timed splash screen that will display the project name, group members and a loading bar. After the loading bar is finished the main program window will populate the screen.

### Randomly Generated Problems

- The program will generate random math questions that a 1st grader can solve. The math problems must be simple and easy to solve. Math questions must only include addition and subtraction operations. Questions must only result in answers that are unsigned integers.
- VALID RANDOMLY GENERATED PROBLEMS:
    - ✓ 5 + 4
    - ✓ 2 + 2
    - ✓ 9 - 3
    - ✓ 7 - 4

### User Input

- Create a numpad for user input. The numpad must include the following numbers: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 as well as a "enter" and "delete" button. Purpose of the numpad is to create an easy method of user input to enter the answers for the randomly generated math questions.

### Data Validation

- Prohibit user from entering in any 'unusual' input that could possibly break the program and include any form of error catching.
- EXAMPLES OF 'UNUSUAL' INPUT COULD INCLUDE:
    - ✓ Chars/strings when an integer input is included
    - ✓ Whitespace input
    - ✓ Symbols, operators

### Limit Attempts

- Limit the amount of attempts the user has to input the correct answer. User will have 3 attempts at a question, if they exceed their attempt count then they will be given the option to move onto the next question or try again.

### Visual/Audio Feedback

- *Visual Feedback*: Display text that tells the user their answer was correct. Alternatively, display text that tells the user if their answer was incorrect. Feedback must be encouraging and kid friendly.
    - ➢ EXAMPLE FEEDBACK:
        - ✓ "Good job, you got the answer rights!"
        - ✓ "Oops, wrong answer! Try again!"
    - o Provide a fun visual background.
- *Audio Feedback* – include the following:
    - ✓ Background music in the application.
    - ✓ Input SFX when the user presses buttons.

- ✓ Correct answer SFX – ding ding ding sound effect.
- ✓ Incorrect answer SFX – buzzer sound effect.

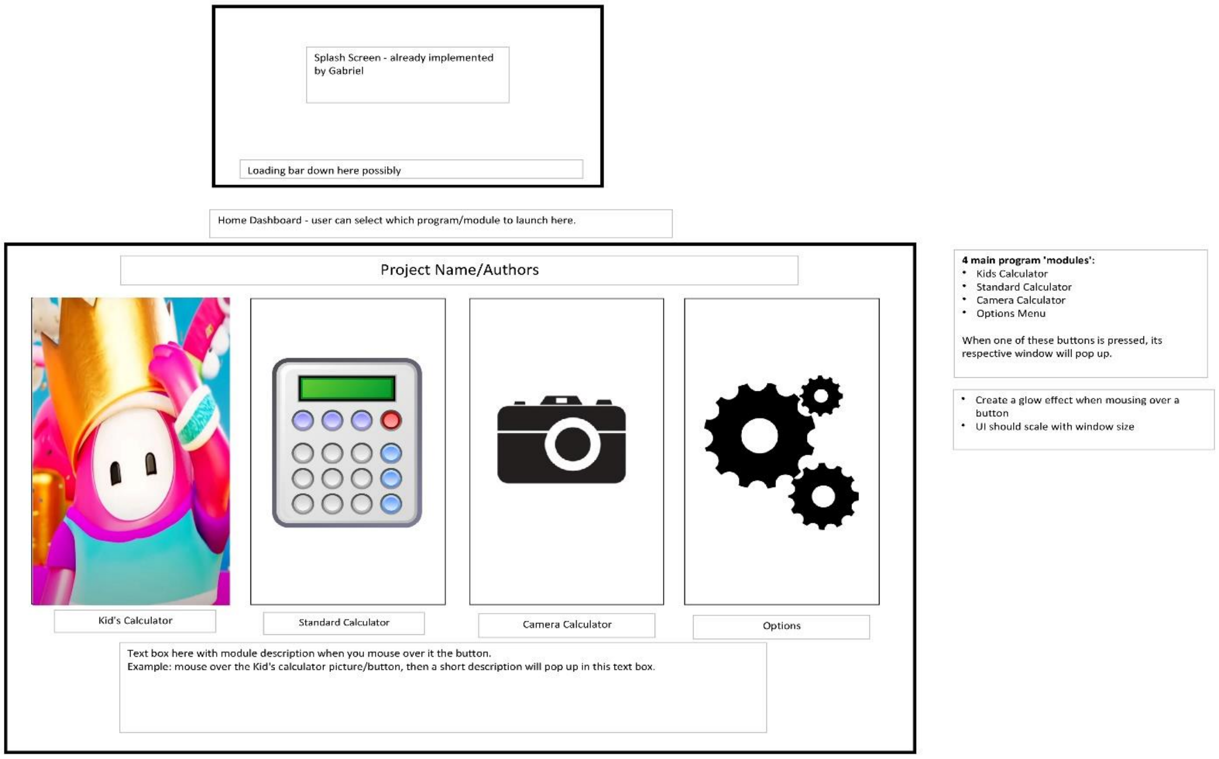
### Picture Assisted Problem Solving

- Target audience is aimed at 1st graders, they may struggle with reading numbers. Provide pictures that represent the randomly generated math question.
  - For example: the math question could be "What is 4 + 2"? The application will then display 4 balls + 2 balls, to visually assist the user in counting the images.

### GUI

- Create a simple, modern GUI for the user to navigate. Must limit the number of input/clickable widgets on screen to not confuse the user.
  - ➢ *Python must be the base language*.
  - ➢ Use TKinter to create the GUI
    - ✓ Add styling to all widgets

## UML Diagrams <TBD>


Splash Screen - already implemented by Gabriel
Loading bar down here possibly
Home Dashboard - user can select which program/module to launch here.
Project Name/Authors
Kid's Calculator
Standard Calculator
Camera Calculator
Options

Text box here with module description when you mouse over it the button.
Example: mouse over the Kid's calculator picture/button, then a short description will pop up in this text box.

**4 main program 'modules':**
- Kids Calculator
- Standard Calculator
- Camera Calculator
- Options Menu

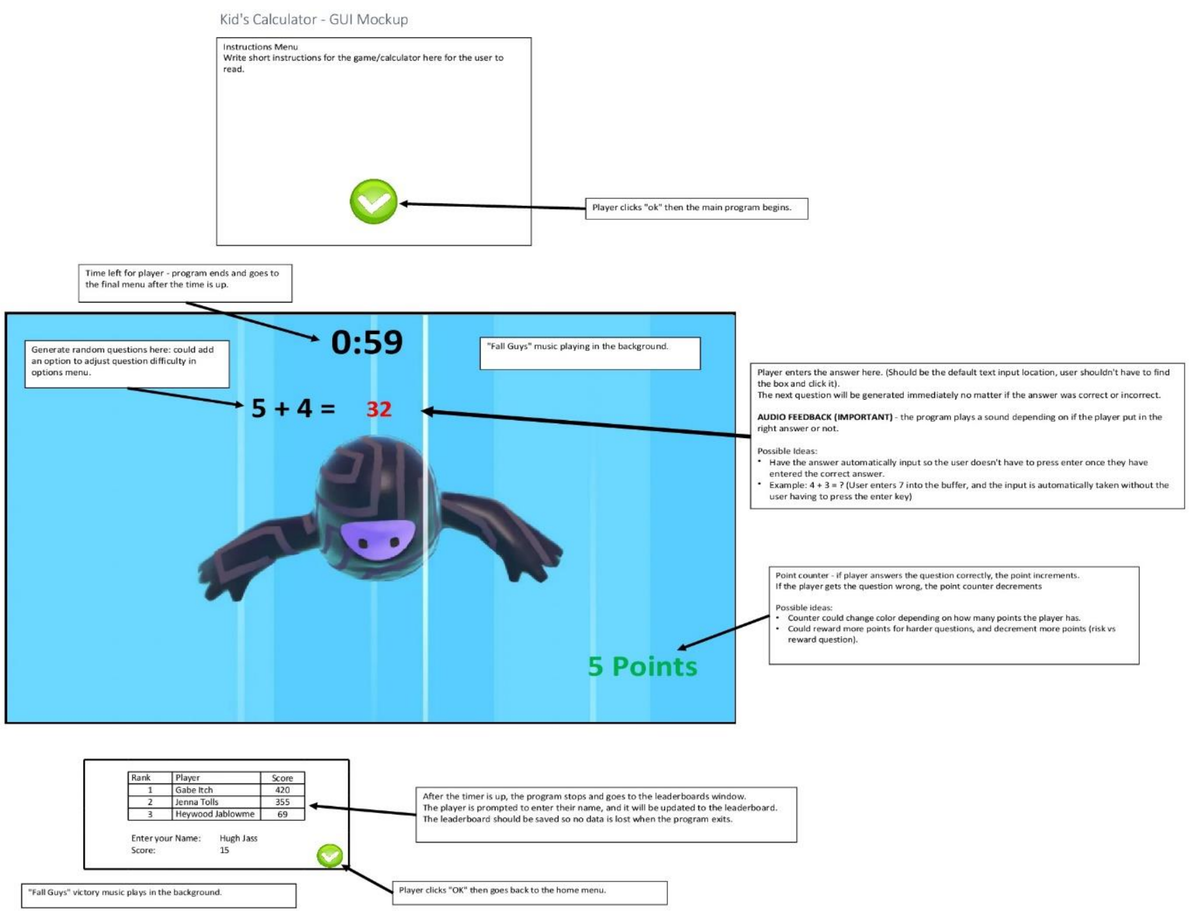
When one of these buttons is pressed, its respective window will pop up.

- Create a glow effect when mousing over a button
- UI should scale with window size

### Use Case Diagram

- Early mockup design of the home page GUI. Originally the application was planned to have 4 features, but these features were decided to be cut out of the final build to keep the application simple for the intended audience.

- Kids Calculator – the main feature. This feature was a simple math game designed for children.
- Standard Calculator (legacy feature that was cut out) – a standard calculator feature that could perform the same mathematical functions any other standard calculator could execute.
- Camera Calculator – This module makes use of the user's web cam to scan a math problem visually then solve it for them.
  - ✓ Used OpenCV and Tesseract to implement this feature.
  - ✓ Not finalized, very early working build.
- Options (legacy feature that was cut out) – An options menu to configure the application based off the user's desires.


Kid's Calculator - GUI Mockup

Instructions Menu
Write short instructions for the game/calculator here for the user to read.

Player clicks "ok" then the main program begins.

Time left for player - program ends and goes to the final menu after the time is up.

Generate random questions here: could add an option to adjust question difficulty in options menu.

"Fall Guys" music playing in the background.

Player enters the answer here. (Should be the default text input location, user shouldn't have to find the box and click it).
The next question will be generated immediately no matter if the answer was correct or incorrect.

**AUDIO FEEDBACK (IMPORTANT)** - the program plays a sound depending on if the player put in the right answer or not.

Possible Ideas:
- Have the answer automatically input so the user doesn't have to press enter once they have entered the correct answer.
- Example: 4 + 3 = ? (User enters 7 into the buffer, and the input is automatically taken without the user having to press the enter key)

Point counter - if player answers the question correctly, the point increments.
If the player gets the question wrong, the point counter decrements

Possible ideas:
- Counter could change color depending on how many points the player has.
- Could reward more points for harder questions, and decrement more points (risk vs reward question).

| Rank | Player | Score |
|---|---|---|
| 1 | Gabe Itch | 420 |
| 2 | Jenna Tolls | 355 |
| 3 | Heywood Jablowme | 69 |

Enter your Name: Hugh Jass
Score: 15

After the timer is up, the program stops and goes to the leaderboards window.
The player is prompted to enter their name, and it will be updated to the leaderboard.
The leaderboard should be saved so no data is lost when the program exits.

"Fall Guys" victory music plays in the background.

Player clicks "OK" then goes back to the home menu.

- This was an early mockup of the kid's calculator game. The final build still looks like this mockup; however, the points, leaderboard and timer features were cut out. We decided to cut out these features because we did not want to rush or stress the audience when trying to learn math.

## Authors

Gabriel Magallanes

gabe695@csu.fullerton.edu

https://github.com/GabeMags

Mohammad Mirwais

mmirwais (Mirwais Milyar) (github.com)

Sean Mitchell

smitchell36@csu.fullerton.edu

https://github.com/slimsloth