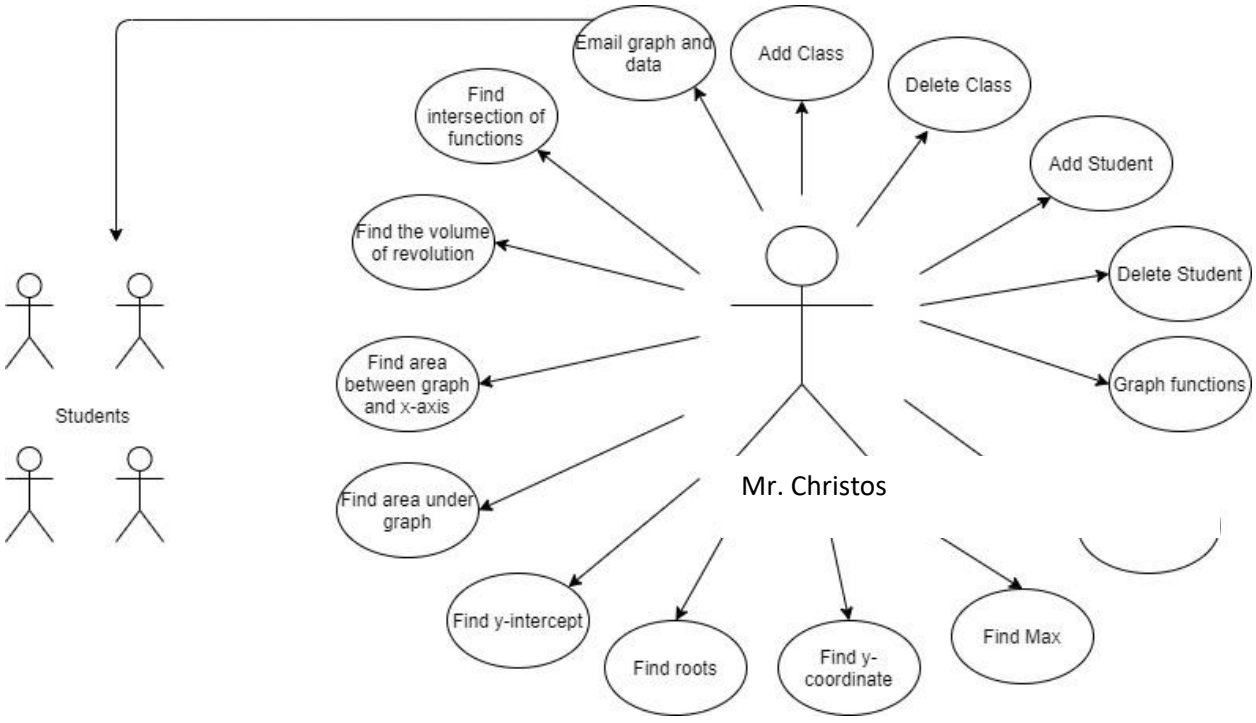
Design

Contents

1. Use cases

This diagram represents the relationship between Mr. Christos, his students and the program I will be implementing.

Figure 1.1: Use cases



2. First visualizations

I drafted first visualizations of the program's windows to see whether they satisfy my clients' needs.

Figure 2.1: "Welcome page" window

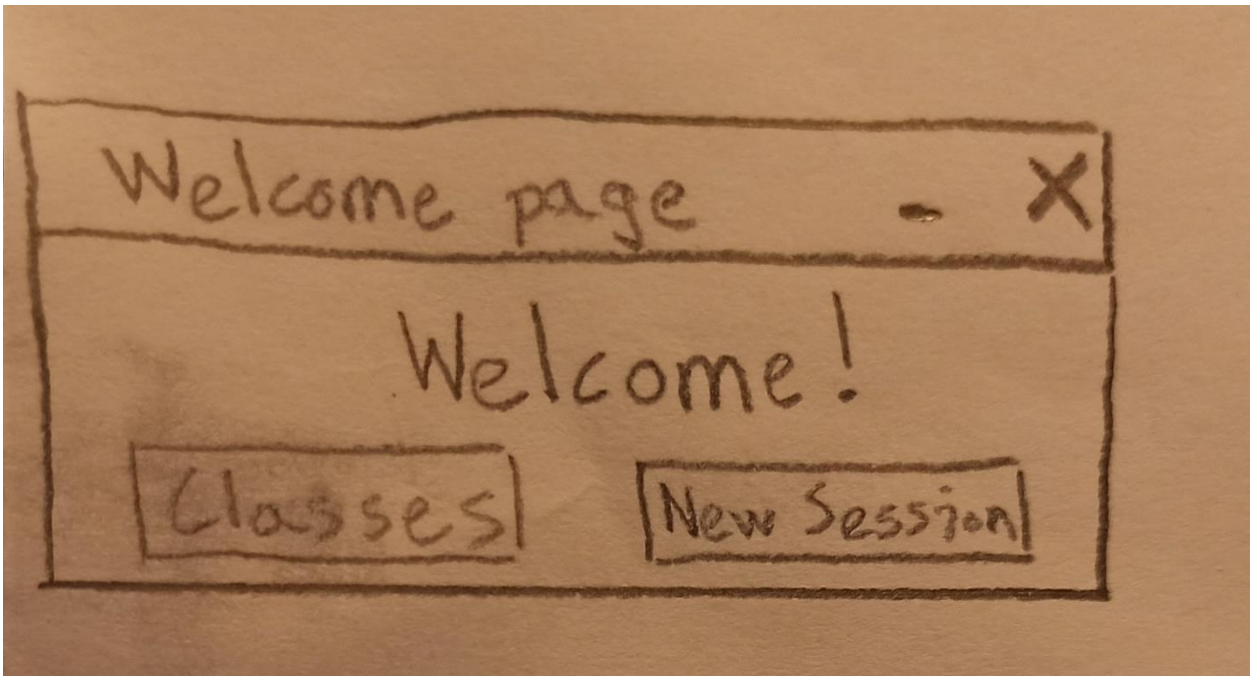


Figure 2.2: "Classes" window

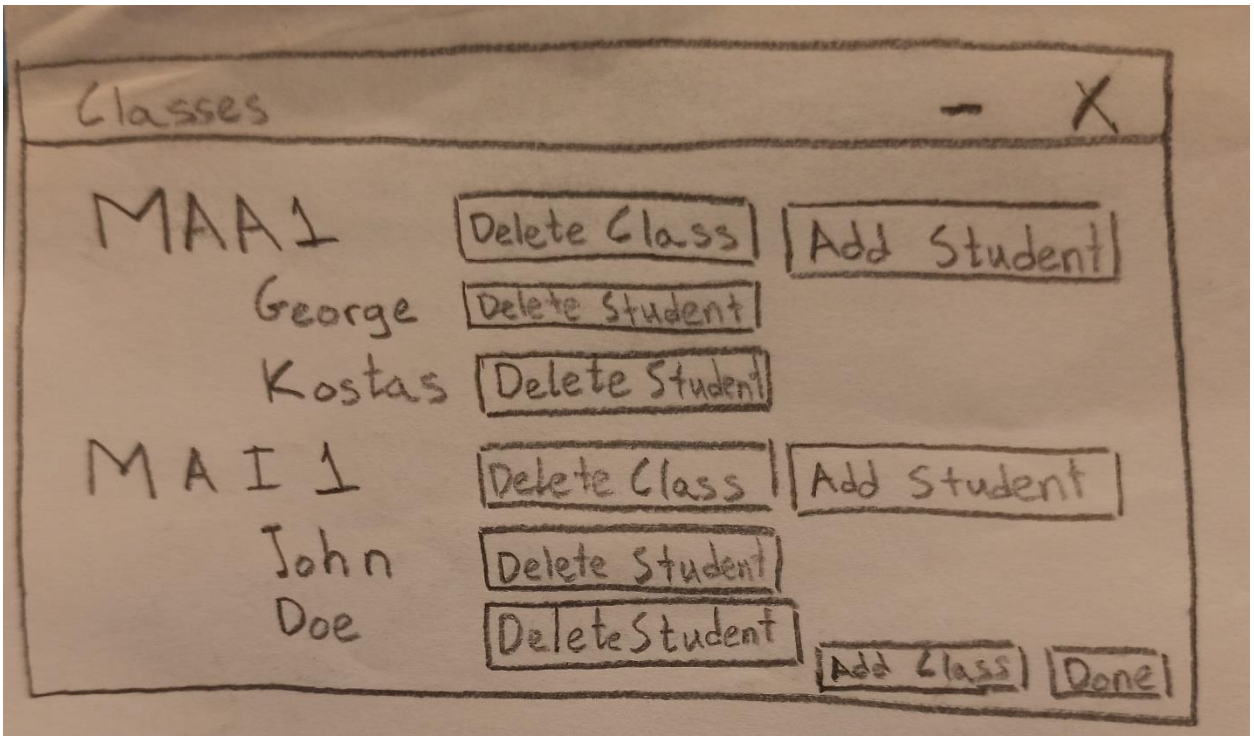


Figure 2.3: "Add Class" window

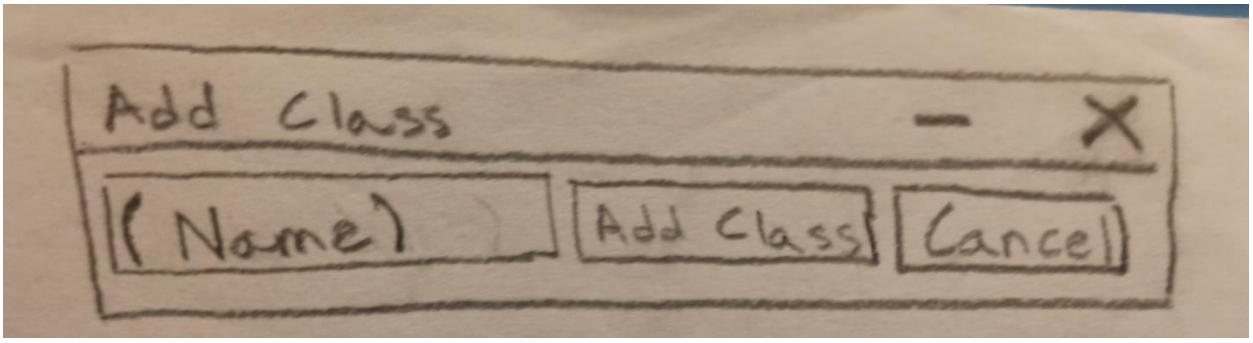


Figure 2.4: "Add Student" window

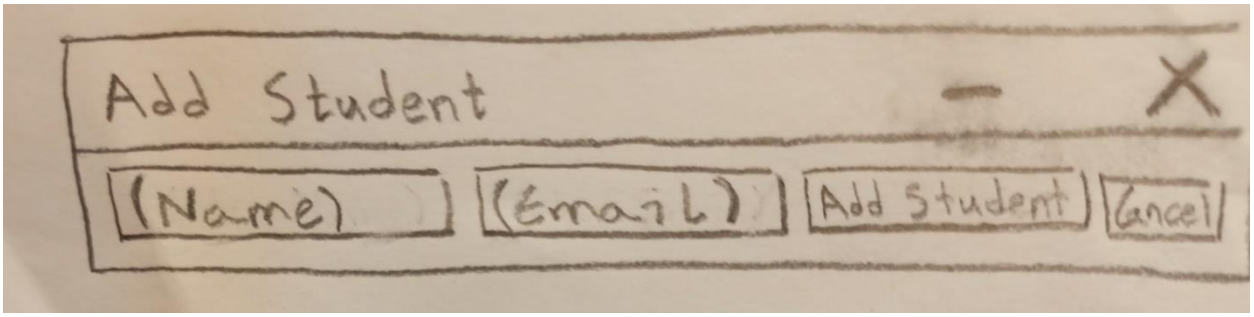


Figure 2.5: "Choose Class" window

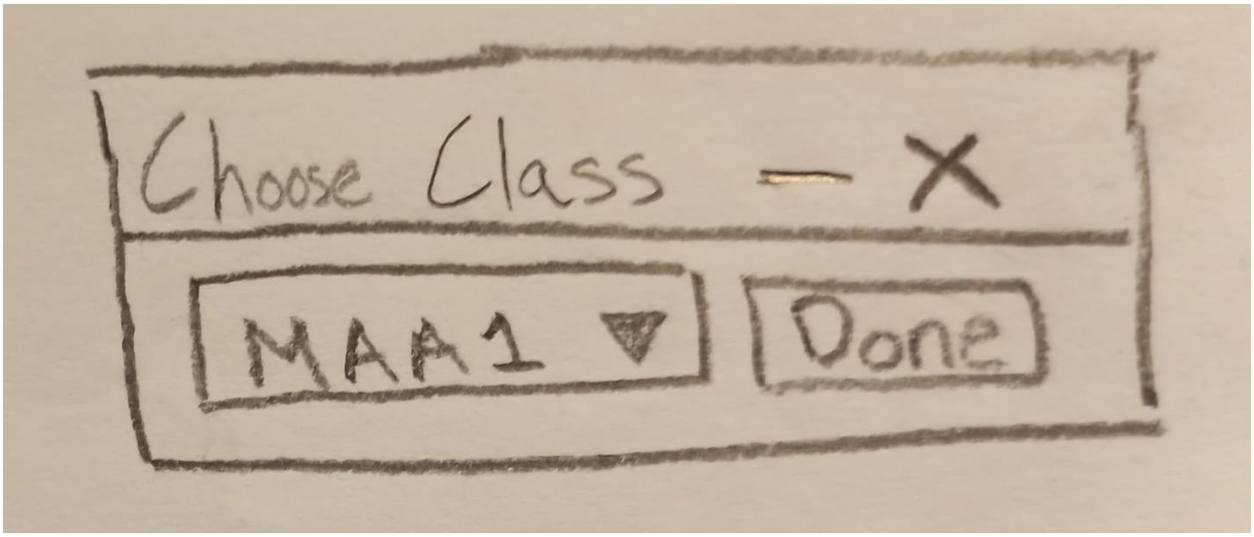


Figure 2.6: "Functions" window

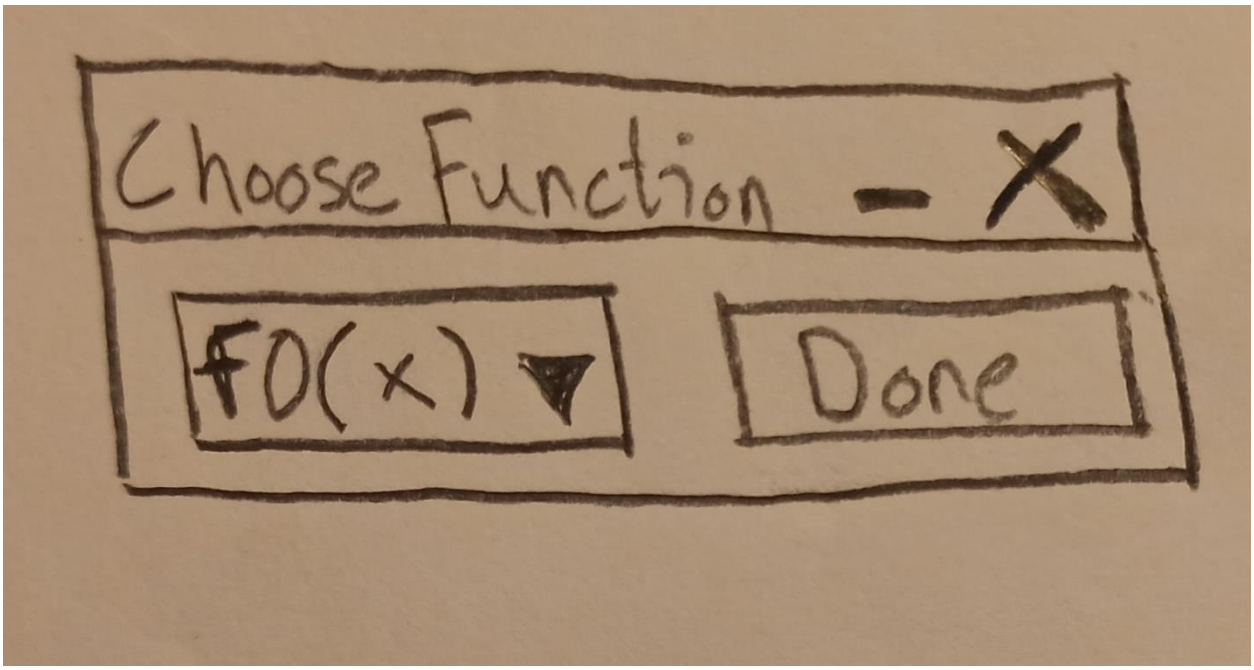


Figure 2.7: "Error window"

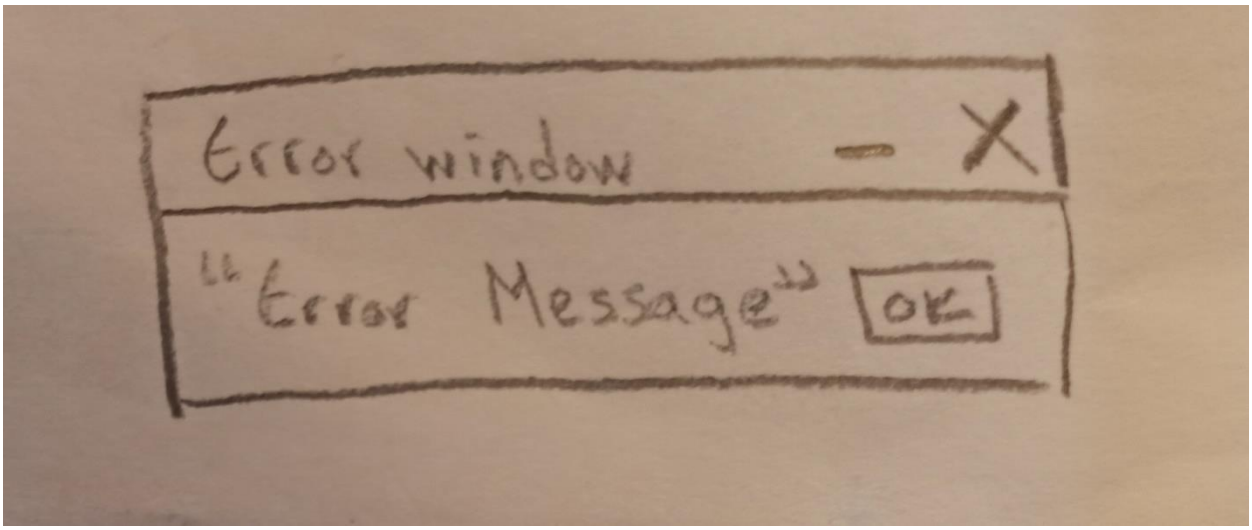


Figure 2.8: "Graph" window

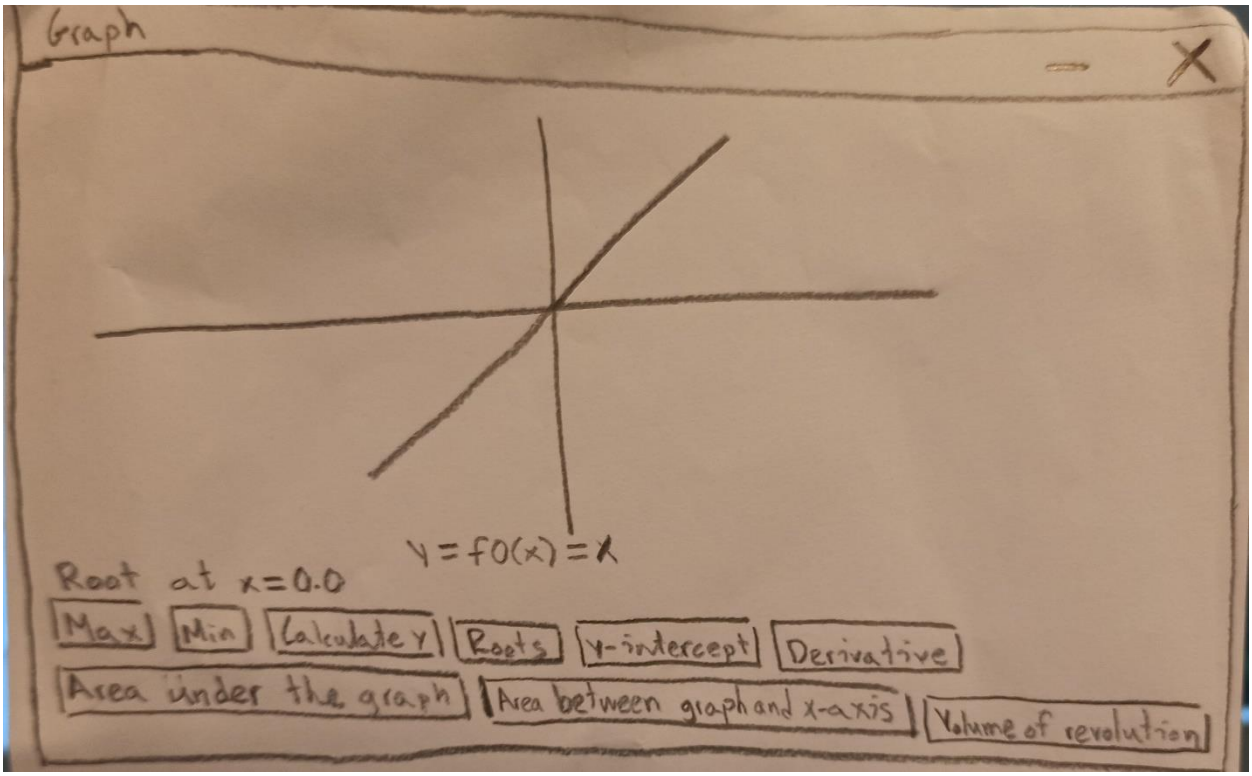
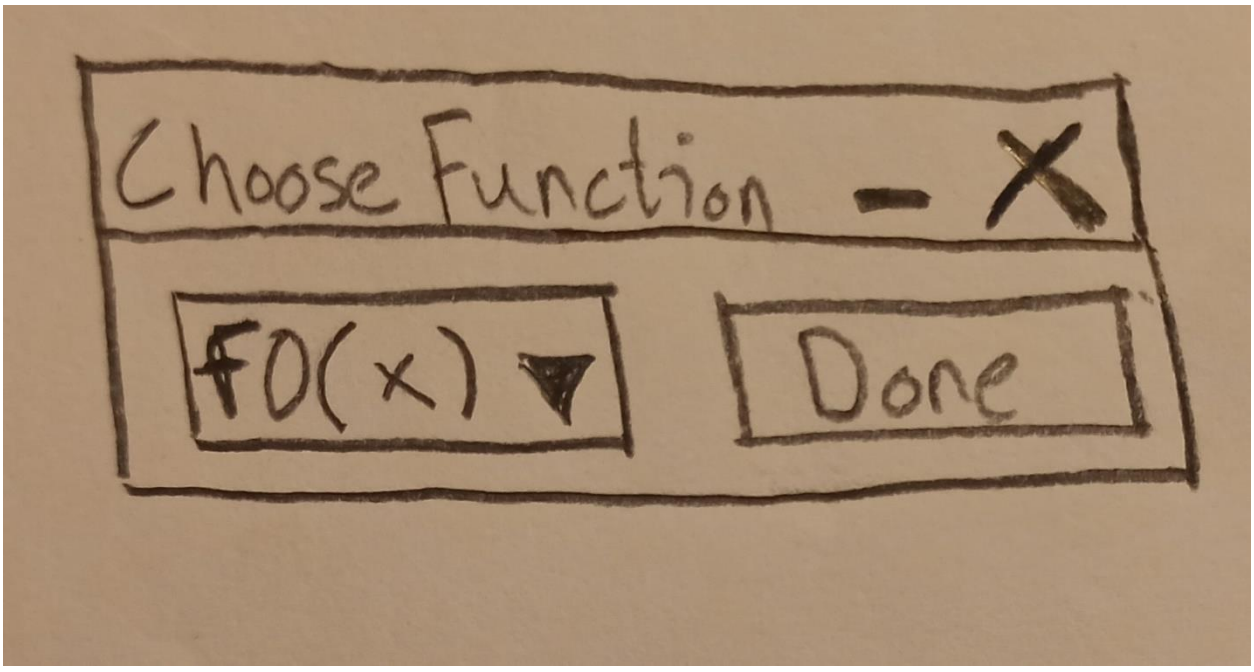


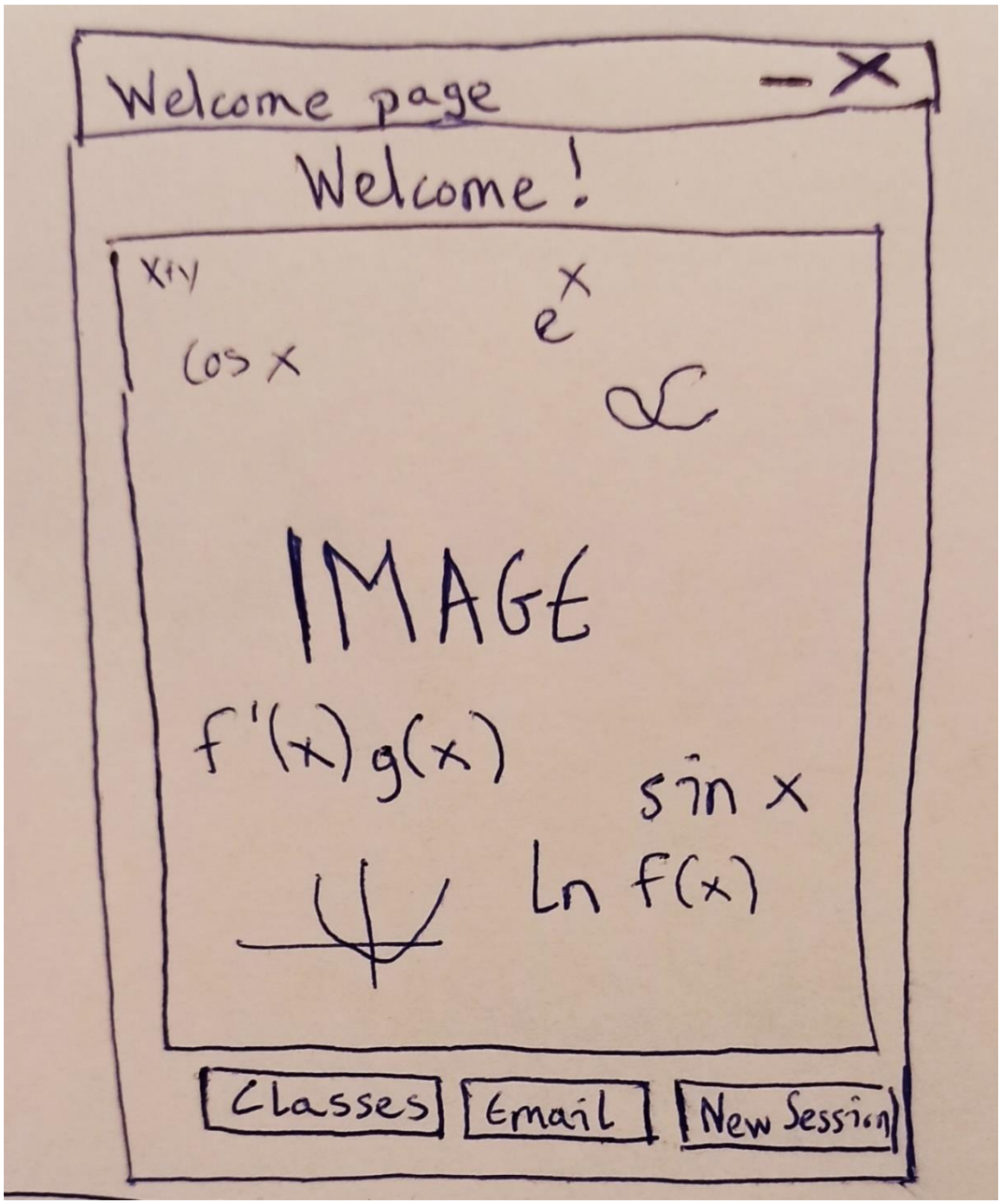
Figure 2.9: "Choose Function" window



3. Final visualizations

After discussion with Mr. Christos [1], we concluded that some changes should be made. Therefore, I redrafted some of the visualizations and ended up with the following:

Figure 3.1: "Welcome page" window



[1] see Appendix B

Figure 3.2: "Classes" window

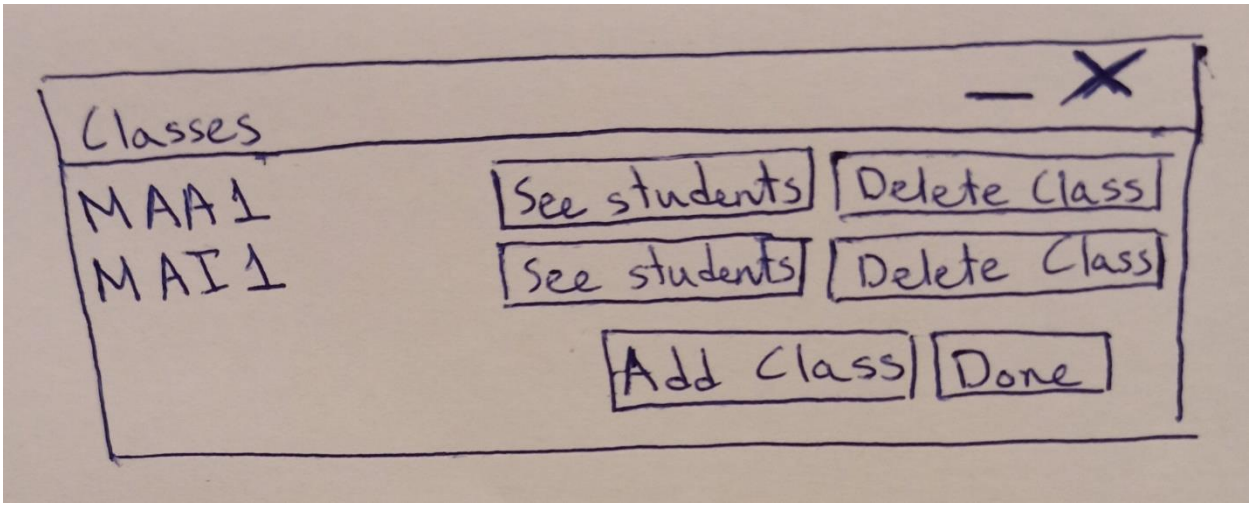


Figure 3.3: "Students" window

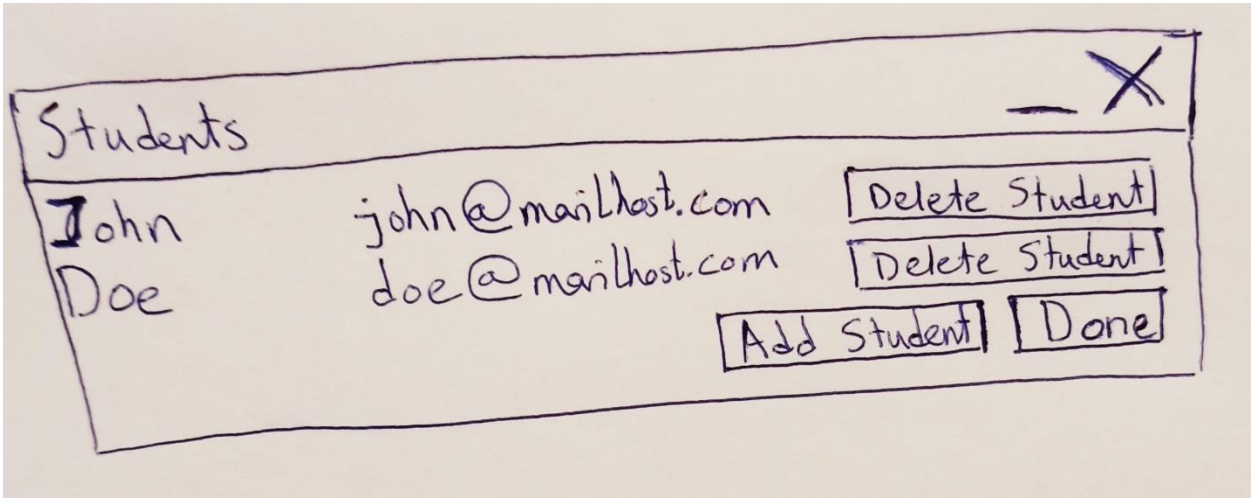


Figure 3.4: "Add Class" window

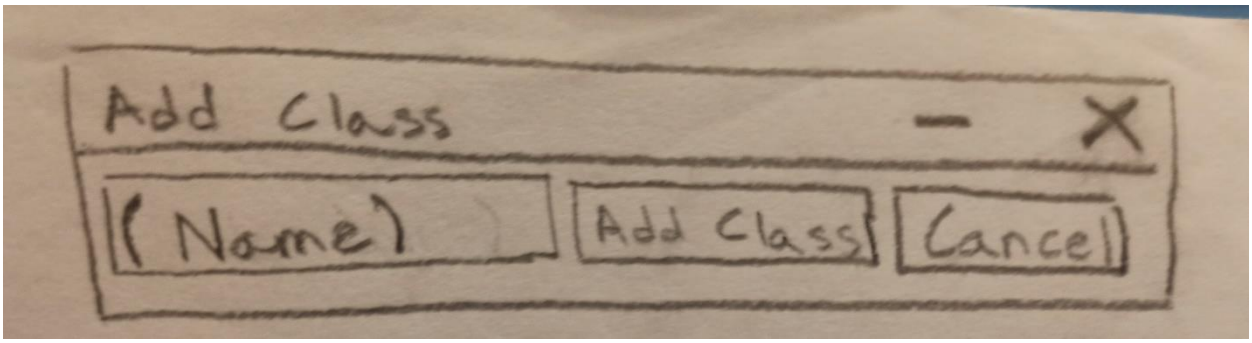


Figure 3.5: "Add Student" window

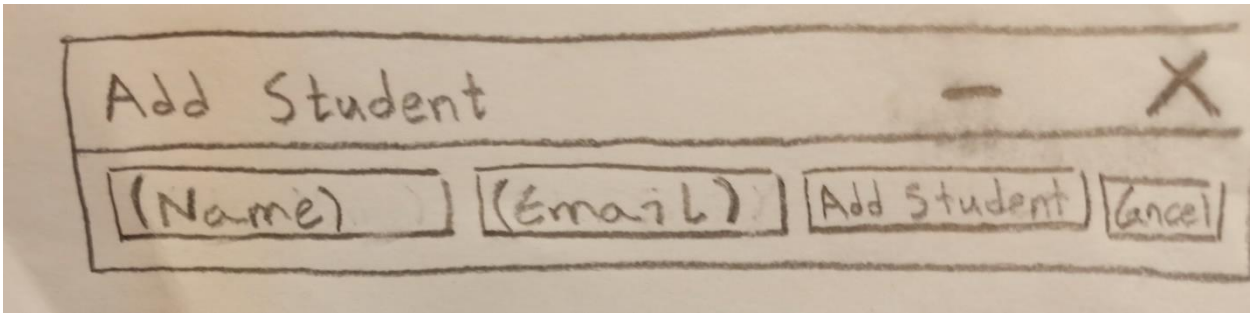


Figure 3.6: "Choose Class" window

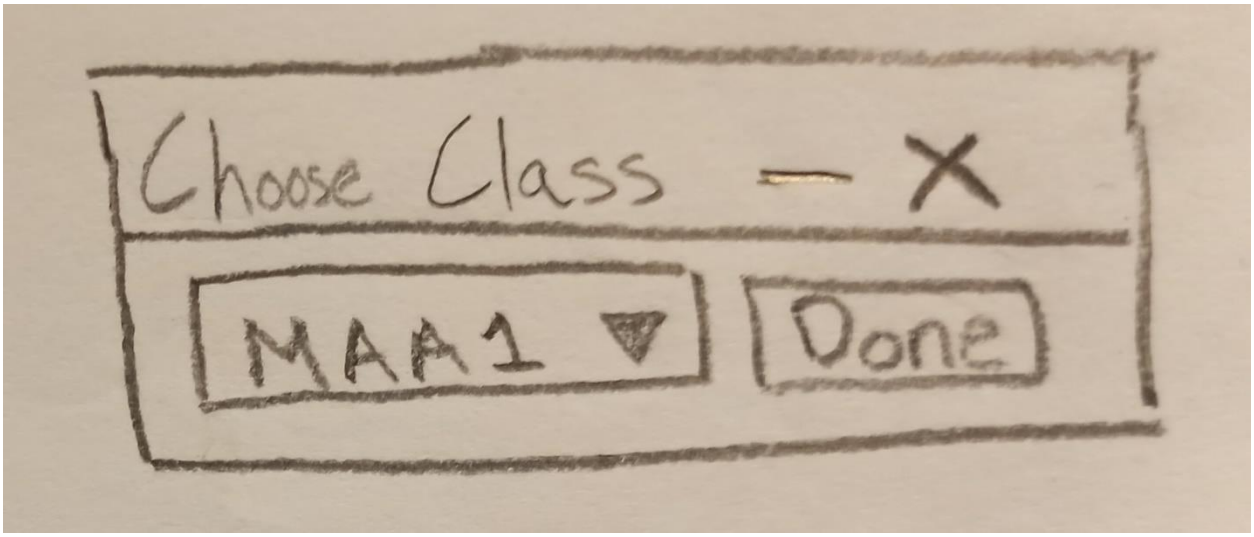


Figure 3.7: "Functions" window

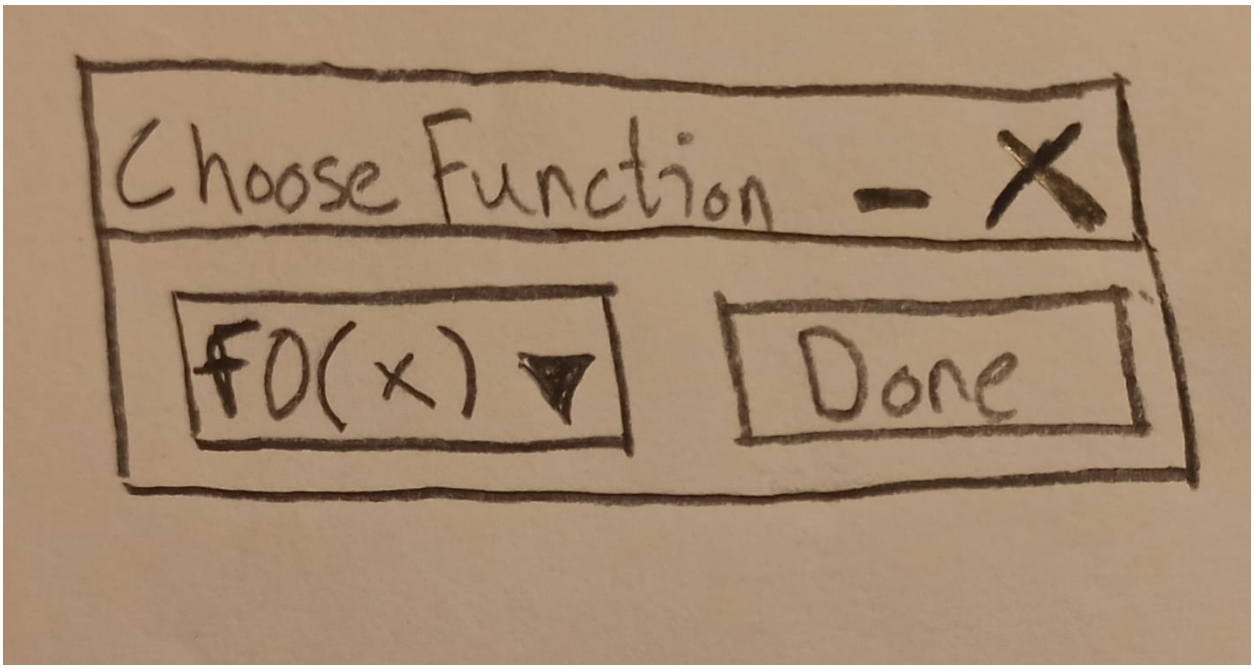


Figure 3.8: "Error window"

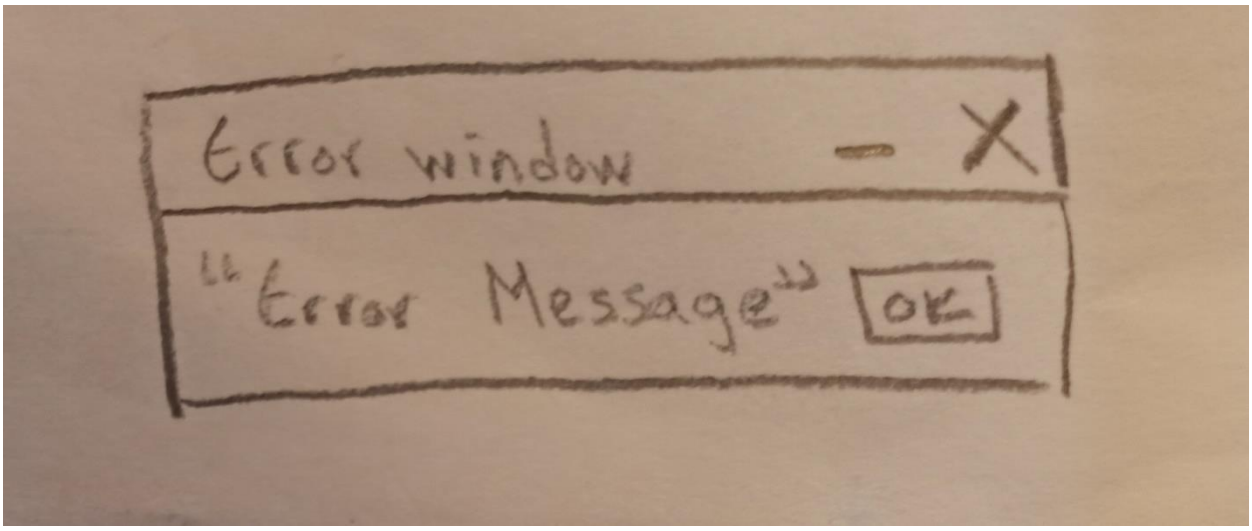


Figure 3.9: "Graph" window

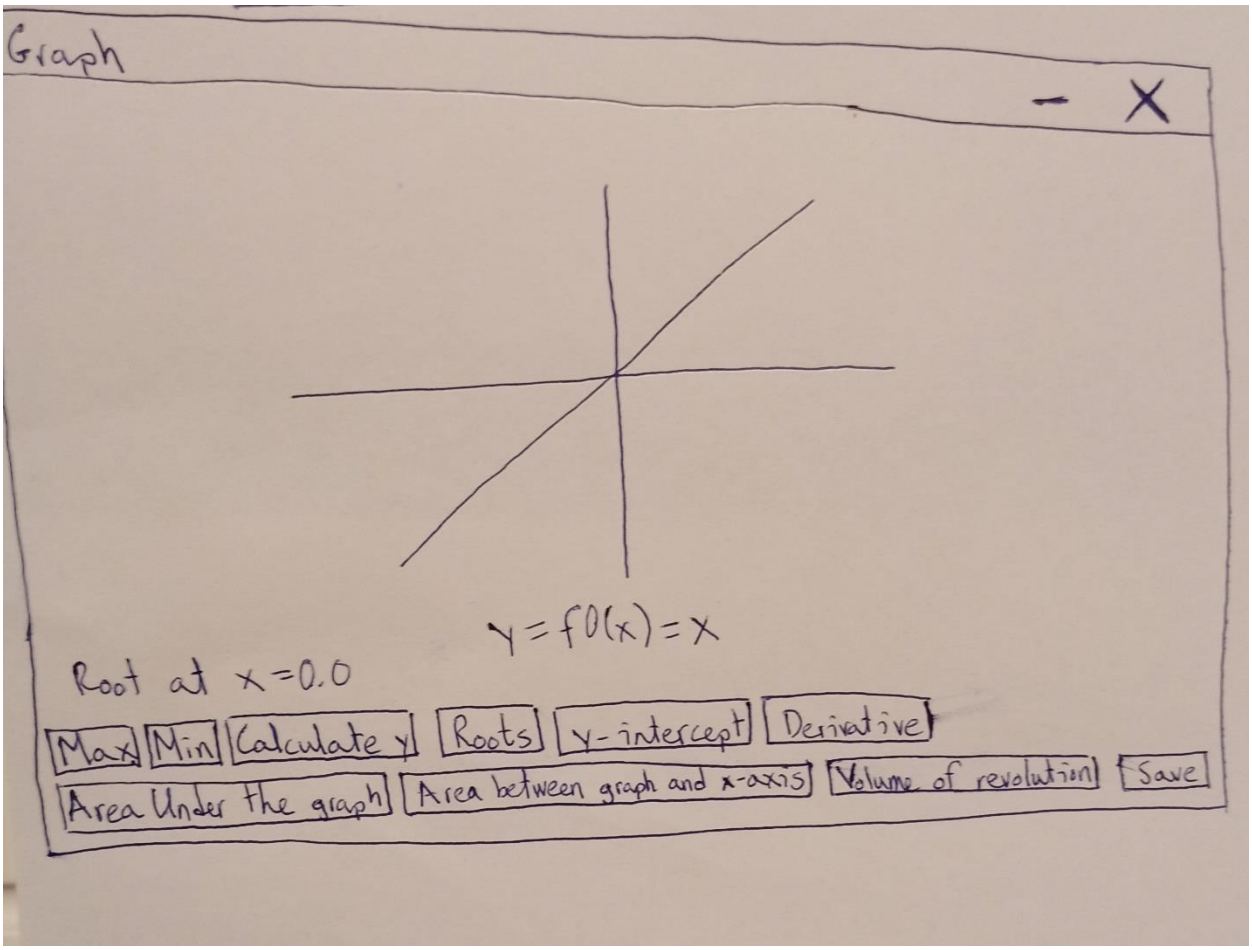
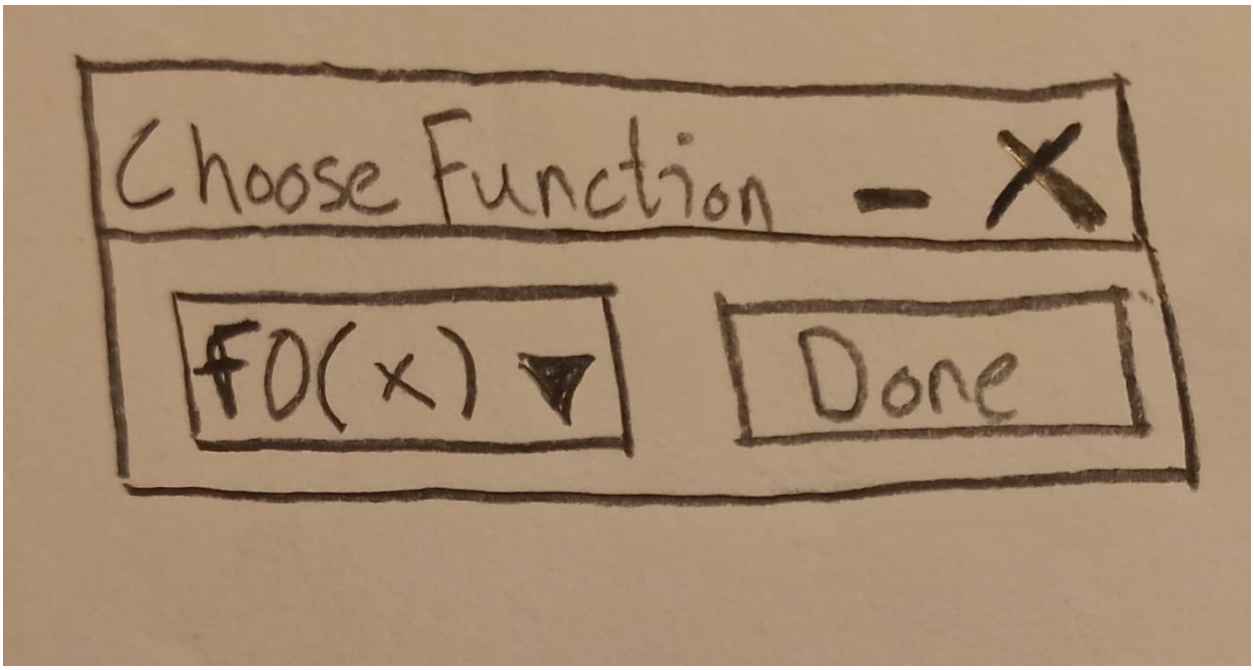


Figure 3.10: "Choose Function" window



4. Data types

The table summarizes how I will be using different data types.

Figure 4.1: Data types

Data Type	Use
Primitive data types and strings	To keep corresponding data types (eg int for number of points)
Linked List	To store classes and students They allow quick insertion and deletion
Array	To pass multiple variables as method arguments
Stage, Scene, Button etc. (from javafx)	To create a dynamic, user-friendly GUI
Function (from mathXParser)	To convert a user input into a defined, callable function

5. Flowcharts

The flowcharts show the menus and different options that the user will have in every case.

Figure 5.1: Method Main

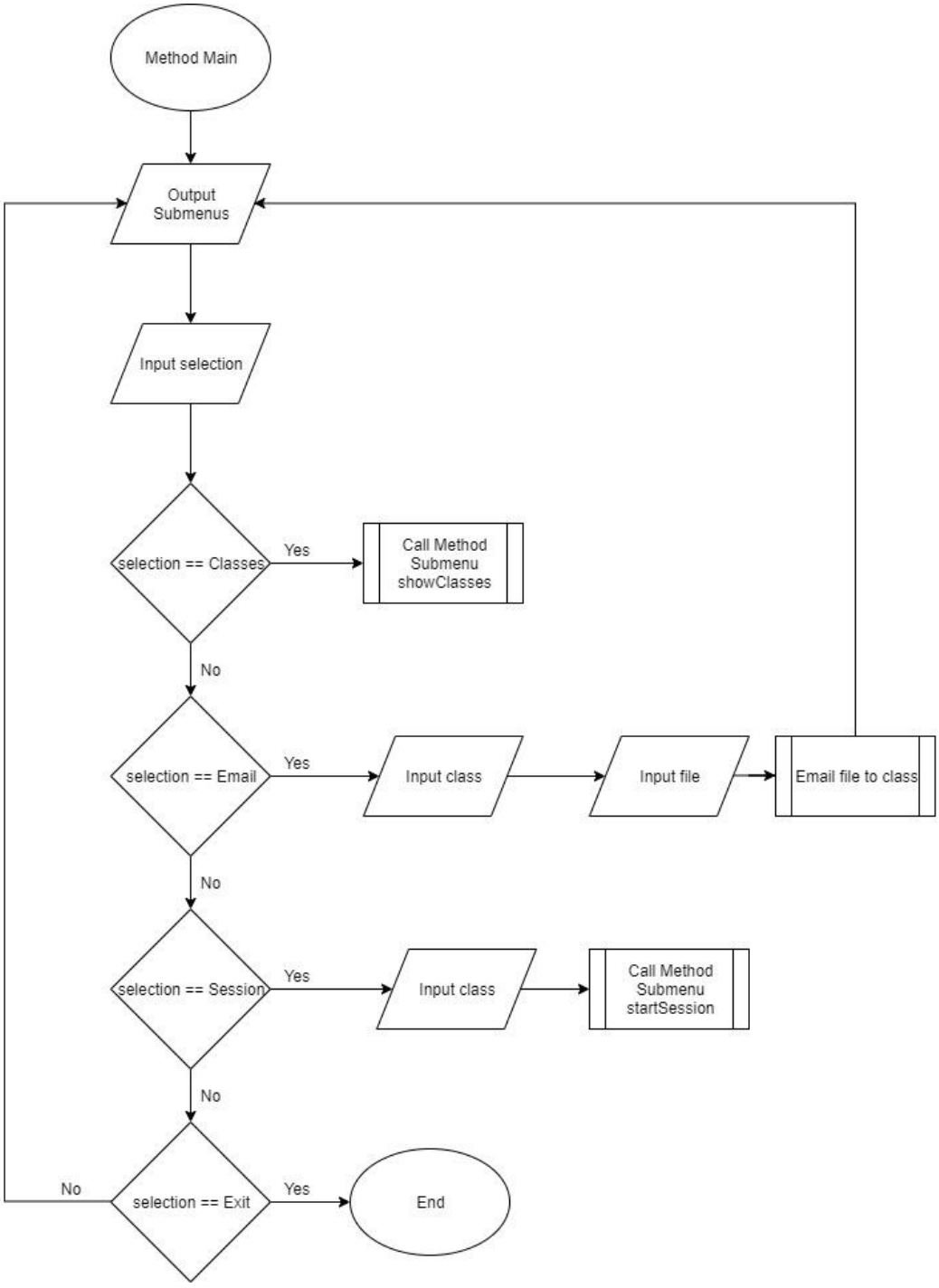


Figure 5.2: Method Submenu showClasses

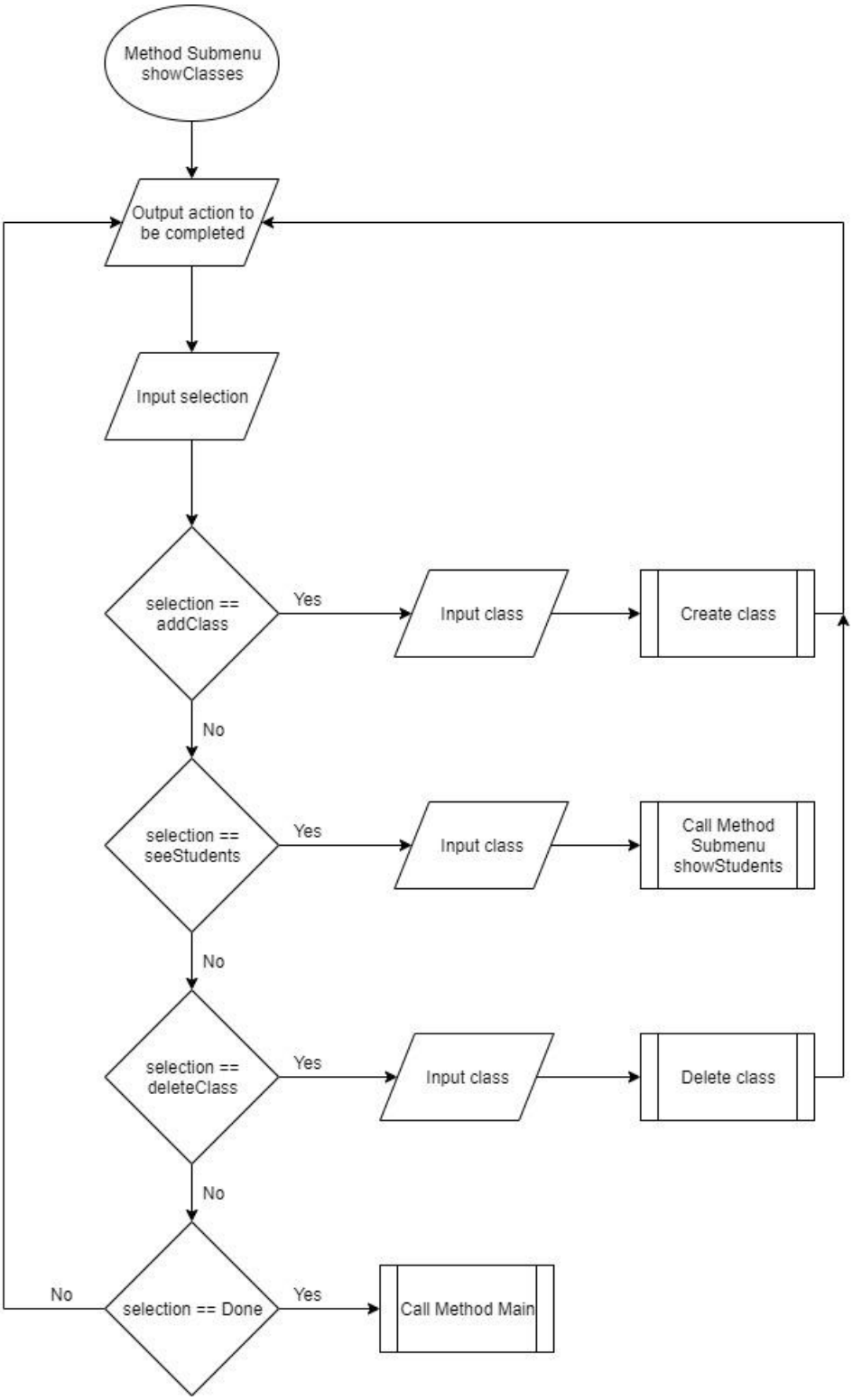


Figure 5.3: Method Submenu showStudents

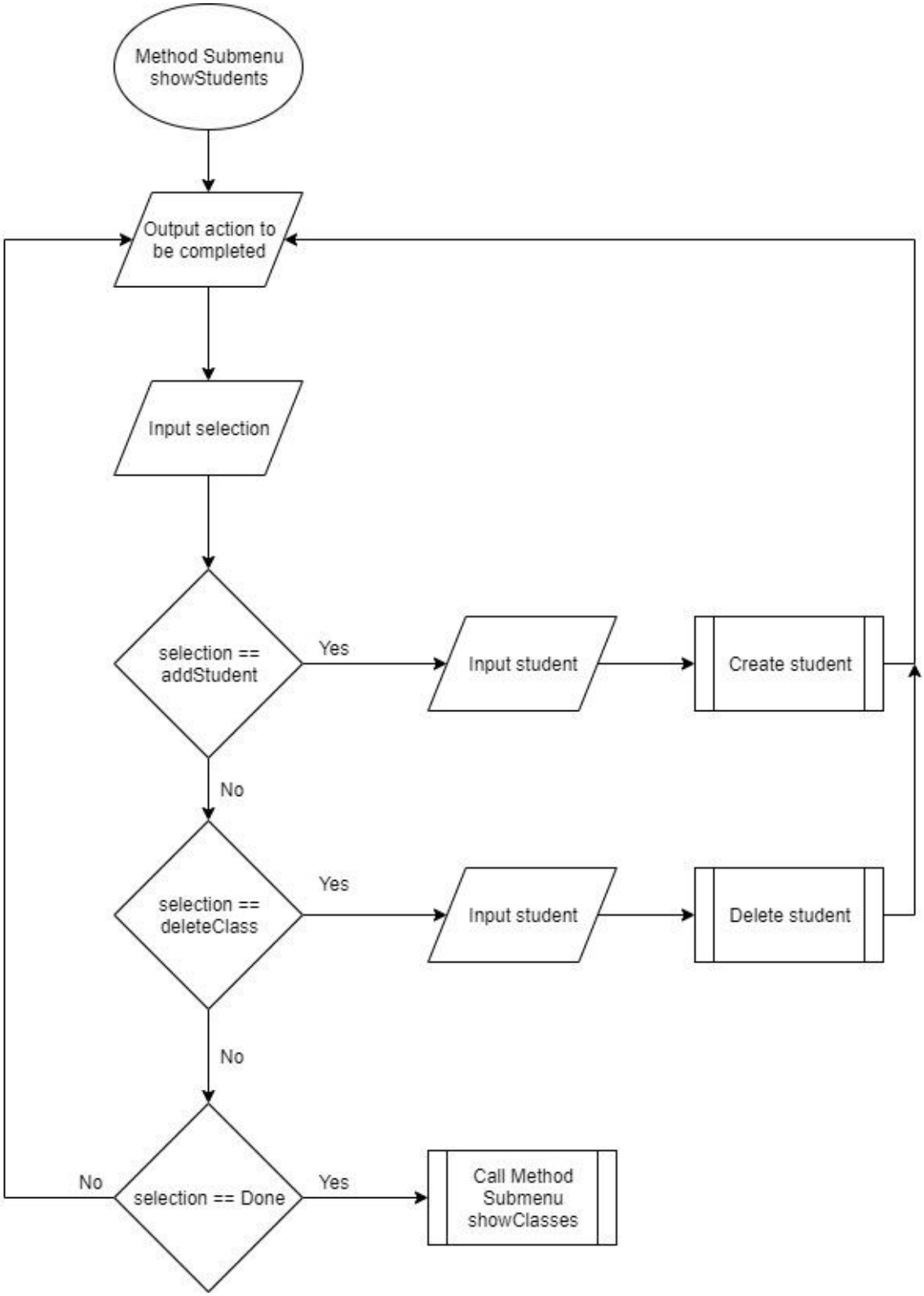


Figure 5.4: Method Submenu startSession

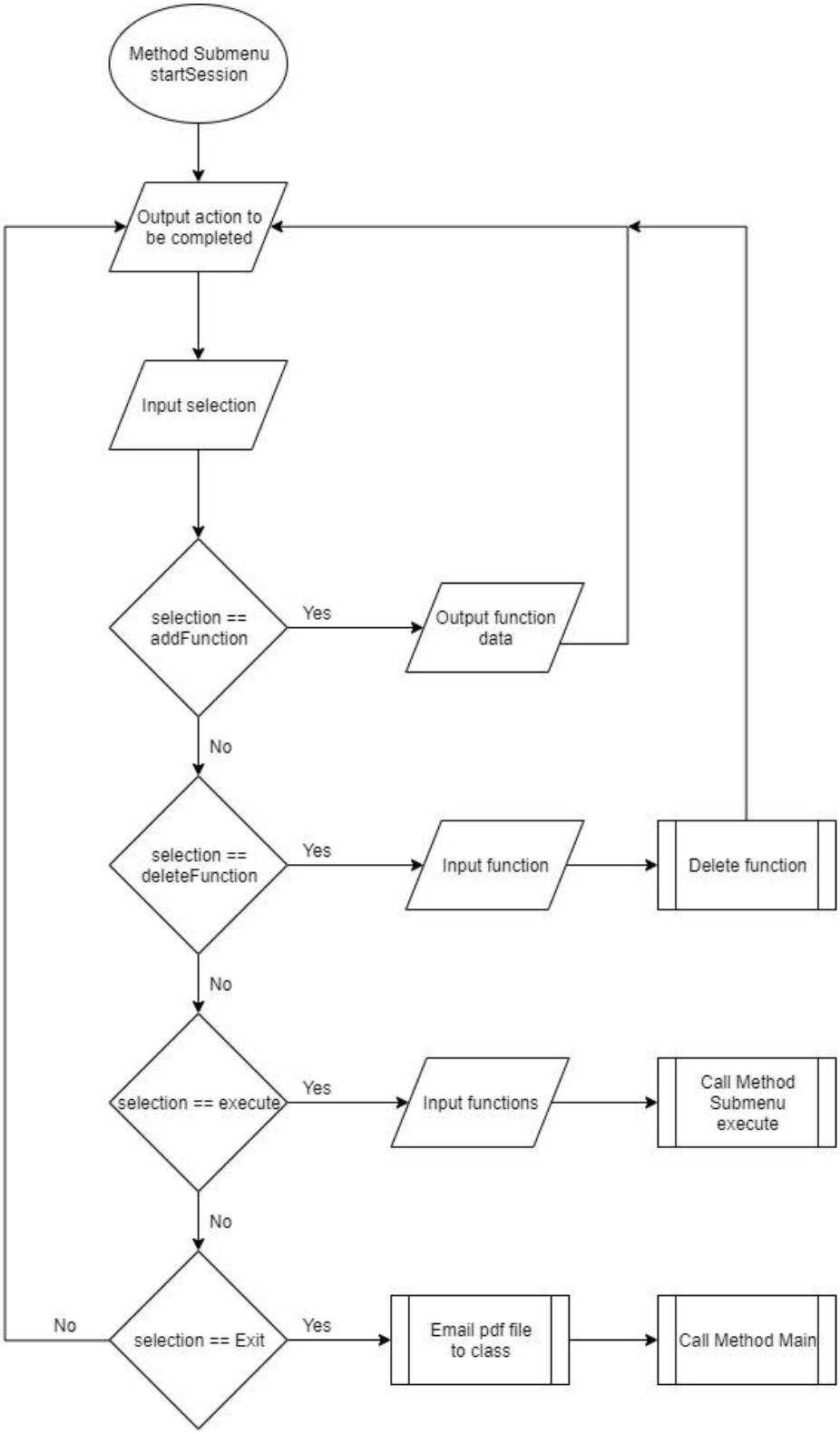


Figure 5.5a: Method Submenu execute

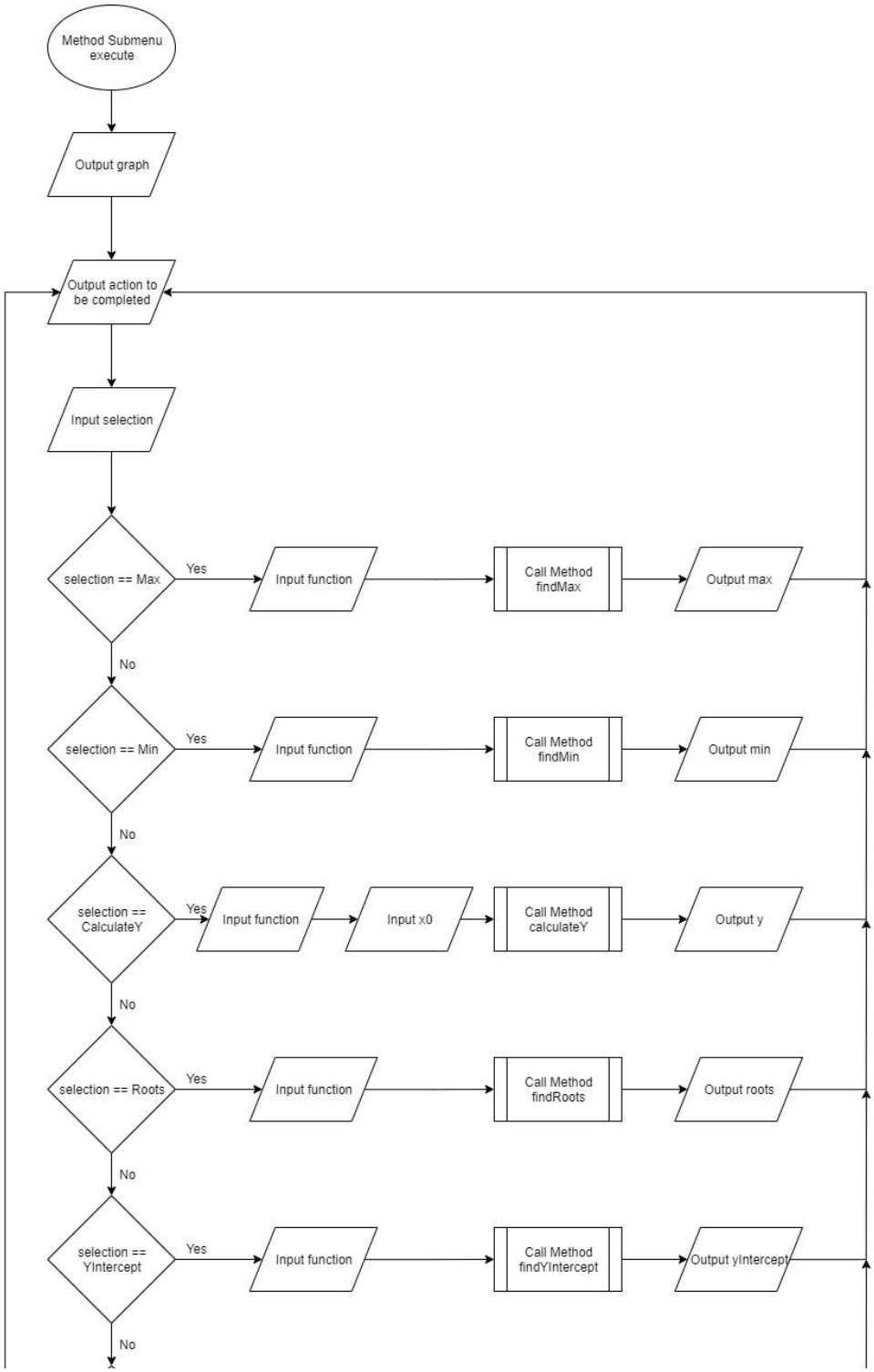
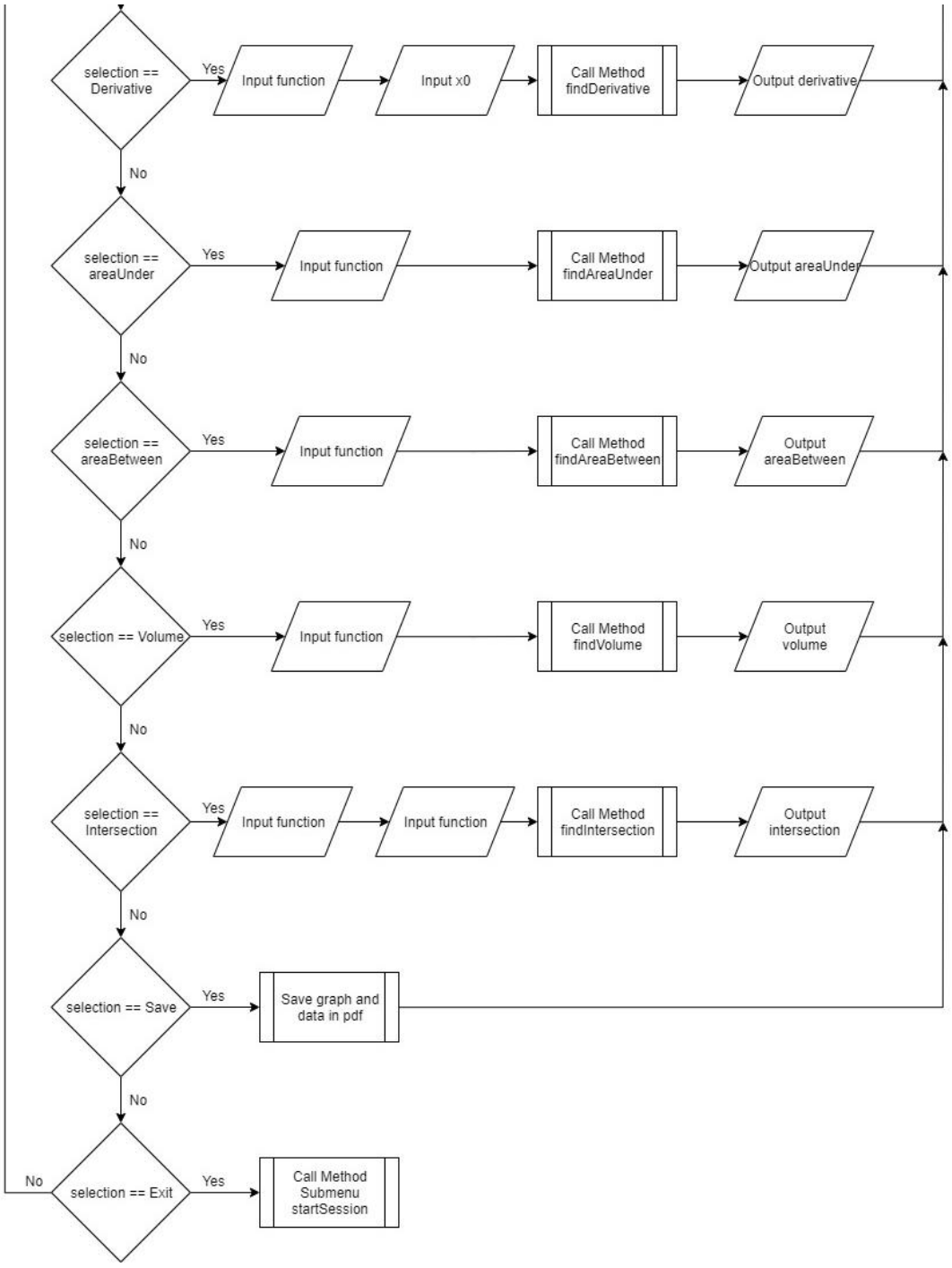


Figure 5.5b: Method Submenu execute (Continued)



6. System flowcharts

These diagrams are concerned with how the program stores information and connects to hardware.

Figure 6.1: Main

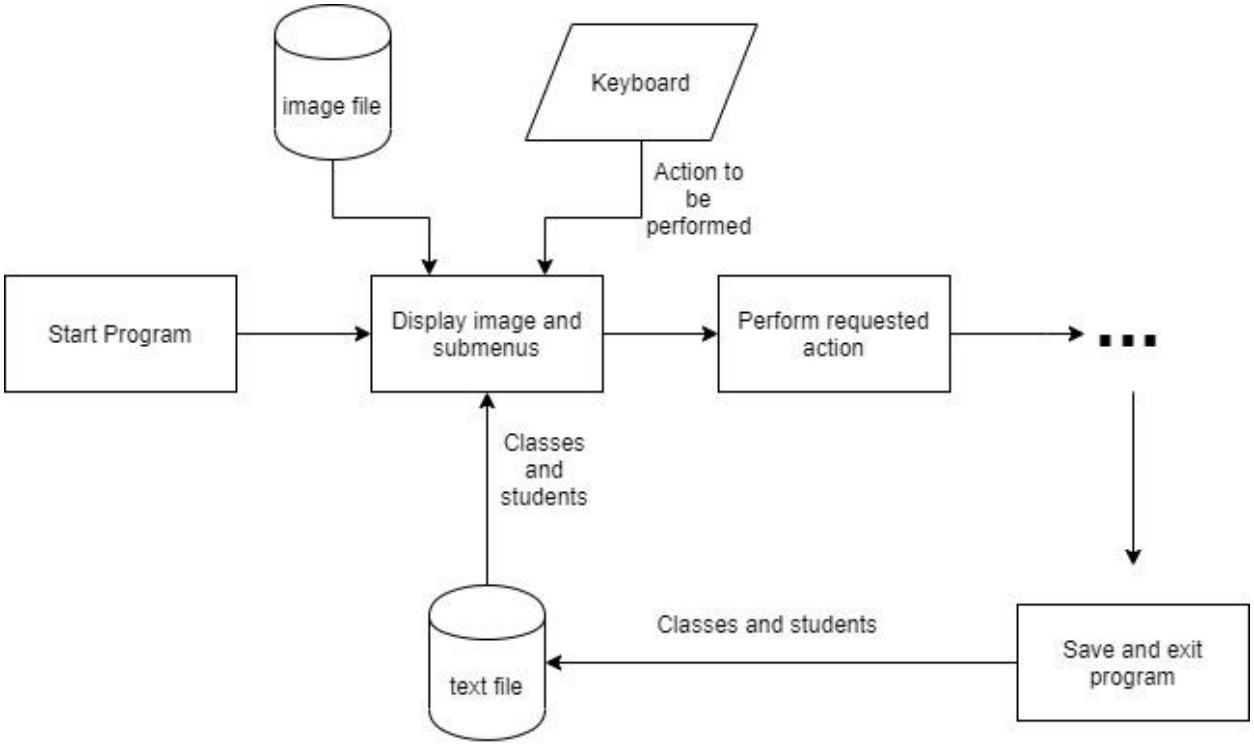
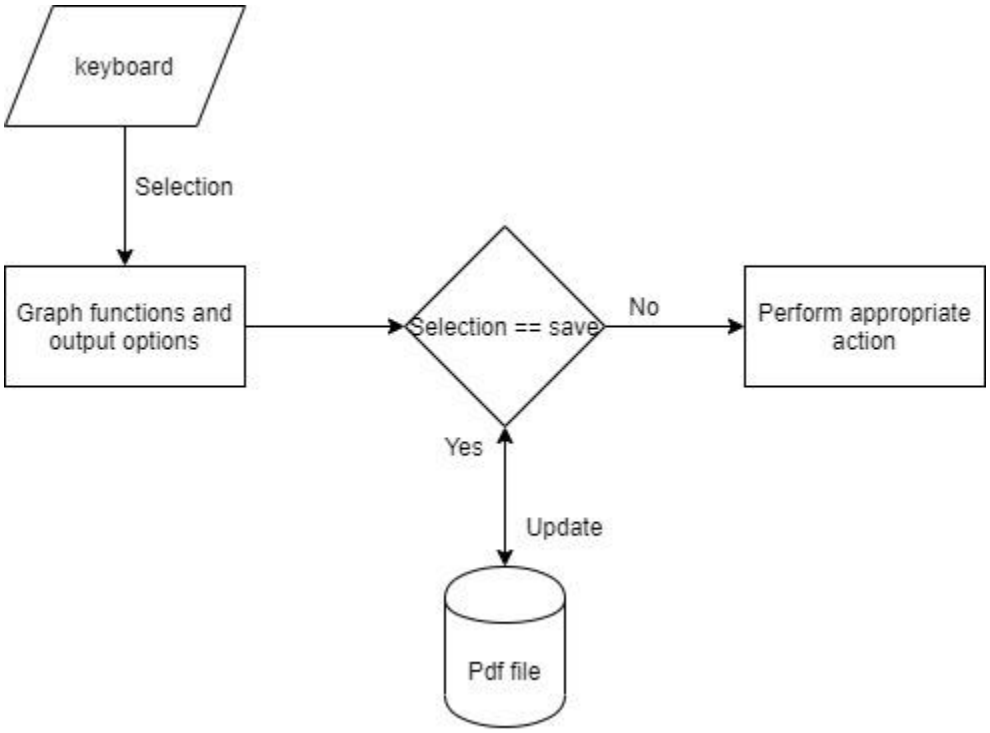


Figure 6.2: Graph



7. Pseudocode

The pseudocode of some complex algorithms is presented below.

- findMax precondition: a given plotted function within a range
- findMax postcondition: the coordinates local maxima are printed

Figure 7.1: findMax

```
//F is the function
//LOWER is the lower bound of the range
//UPPER is the upper bound of the range
method findMax(F, LOWER, UPPER)
    DIFF = (UPPER - LOWER)/20000
    YPREV = F(LOWER)
    YCUR = F(LOWER + DIFF)
    if F(LOWER) > F(LOWER+DIFF) then
        output ("(" + LOWER + ", " + F(LOWER) + ")")
    end if
    D=LOWER
    loop while(D<=UPPER-DIFF)
        YNEXT = F(D+DIFF)
        if(YCUR>YPREV && YCUR>YNEXT) then
            MAX_Y = YPREV
            MAX_X = D-DIFF
            D2 = D-DIFF+(DIFF/20000)
            loop while (D2<=D+DIFF)
                TEMP = F(D2)
                if TEMP > MAX_Y then
                    MAX_Y = TEMP
                    MAX_X = D2
                else
                    break
                end if
                D2 = D2 + DIFF/20000
            end loop
            output ("(" + MAX_X + ", " + MAX_Y + ")")
        end if
        YPREV = YCUR
        YCUR = YNEXT
        D = D + DIFF
    end loop
    if F(UPPER) > F(UPPER-DIFF) then
        output ("(" + UPPER + ", " + F(UPPER) + ")")
    end if
end method
```

- findAreaUnderGraph precondition: a given plotted function within a range
- findAreaUnderGraph postcondition: the area under the graph of the function is printed

Figure 7.2: findAreaUnderGraph

```
//F is the function
//LOWER is the lower bound of the range
//UPPER is the upper bound of the range
method findAreaUnderGraph(F, LOWER, UPPER)
    AREA = 0
    DIFF = (UPPER-LOWER)/100000
    PREV = LOWER
    D = LOWER + DIFF
    loop while (D<=UPPER)
        TEMP = F(D)
        AREA = AREA + (PREV+TEMP)*DIFF/2
        PREV = TEMP
        D = D + DIFF
    end loop
    output AREA
end method
```

- findRoots precondition: a given plotted function within a range
- findRoots postcondition: the values of x for which f(x)=0 are printed

Figure 7.3a: findRoots

```
method abs(d)
    if d>=0 then
        return d
    else
        return -d
end method

//F is the function
//LOWER is the lower bound of the range
//UPPER is the upper bound of the range
method findRoots(F, LOWER, UPPER)
    ROOT_FOUND = false
    DIFF = (UPPER-LOWER)/20000
    YPREV = F(LOWER)
    D = LOWER
    loop while (D<=UPPER)
        Y = F(D)
        if (YPREV<0 && Y>0) || (YPREV>0 && Y<0) then
            ROOT_FOUND = true
            CLOSEST_TO_ZERO = abs(YPREV)
            X = D - DIFF
            D2=D-DIFF
            loop while(D2<=D)
                TEMP = F(D2)
                if abs(TEMP) < CLOSEST_TO_ZERO then
                    CLOSEST_TO_ZERO = abs(TEMP)
                    X = D2
                end if
                D2 = D2 + DIFF/20000
            end loop
            output X
        end if
        YPREV = Y
        D = D + DIFF
    end loop
    YPREV = F(LOWER)
    YCUR = F(LOWER+DIFF)
    D = LOWER + DIFF
    loop while (D<=UPPER-DIFF)
        YNEXT = F(D+DIFF)
        if (YCUR > YPREV) && (YCUR > YNEXT) && (YPREV < 0) then
            CLOSEST_Y = -YPREV
            CLOSEST_X = D-DIFF
            D2 = D - DIFF + DIFF/20000
            loop while (D2<=D+DIFF)
                TEMP = abs(F(D2))
                if (TEMP < CLOSEST_Y) then
                    CLOSEST_Y = TEMP
                    CLOSEST_X = D2
                else
                    break
                end if
                D2 = D2 + DIFF/20000
```

Figure 7.3b: findRoots (Continued)

```
            end loop
            if (CLOSEST_Y < 1e-5) then
                ROOT_FOUND = true
                output CLOSEST_X
            end if
        end if
        YPREV = YCUR
        YCUR = YNEXT
        D = D+DIFF
    end loop
    YPREV = F(LOWER)
    YCUR = F(LOWER+DIFF)
    D = LOWER + DIFF
    loop while (D<=UPPER-DIFF)
        YNEXT = F(D+DIFF)
        if (YCUR < YPREV) && (YCUR < YNEXT) && (YPREV > 0) then
            CLOSEST_Y = YPREV
            CLOSEST_X = D-DIFF
            D2 = D - DIFF + DIFF/20000
            loop while (D2<=D+DIFF)
                TEMP = abs(F(D2))
                if (TEMP<CLOSEST_Y) then
                    CLOSEST_Y = TEMP
                    CLOSEST_X = D2
                else
                    break
                end if
                D2 = D2 + DIFF/20000
            end loop
            if (CLOSEST_Y < 1e-5) then
                ROOT_FOUND = true
                output CLOSEST_X
            end if
        end if
        YPREV = YCUR
        YCUR = YNEXT
        D = D+DIFF
    end loop
    if(!ROOT_FOUND) then
        output "No root in given interval)
    end if
end method
```

8. Class responsibilities

The table summarizes how different classes will be used in the program.

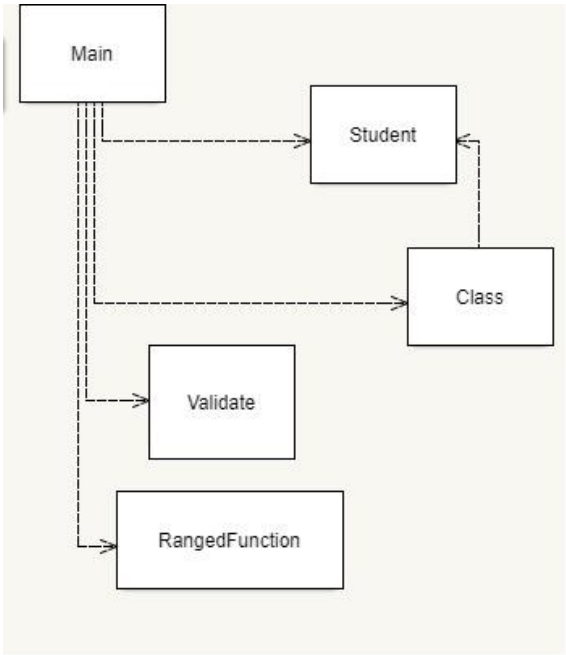
Figure 8.1: Class responsibilities

Main	• Will be responsible for the whole GUI and user input • Will coordinate function of other classes • Will read from and write to file • Will create and email pdf file
Student	• Will represent a student holding their name and email address
Class	• Will represent a class holding its name and its students
Validate	• Will be responsible for doing validation of names and email addresses
RangedFunction	• Will be responsible for holding the functions entered by the user and their range

9. Class connections

The figure shows the relationship between the classes.

Figure 9.1: Class connections



10. UML diagrams

The member-variables and methods that each class will have are presented.

Figure 10.1a: Main

Main
-classes: LinkedList -stgInit: Stage -stgClasses: Stage -stgAddClass: Stage -stgStudents: Stage -stgAddStudent: Stage -stgError: Stage -stgChooseClass: Stage -stgFunctions: Stage -stgGraph: Stage -stgChooseFunction: Stage -stgChoose2ndFunction: Stage

Figure 10.1b: Main (Continued)

-stgGetX: Stage -currentClass: String -prevFileName: String -curFileName: String -curFunc: int -numberOfPoints: int
+main(String[]): void +start(Stage): void +readClasses(): void +btnClassesClicked(): void +chooseClass(boolean): void +getFunctions(): void +btnAddFunctionClicked(VBox): void +btnDeleteFunctionClicked(VBox, int): void +btnExeClicked(VBox, PDDocument): void +execute(RangedFunction[], PDDocument): void +btnSaveFunctionClicked(ScrollPane, PDDocument): void +email(): void +chooseFunction(VBox, RangedFunction[], String): void +choose2ndFunction(VBox, RangedFunction[], int): void +getX(VBox, RangedFunction, String): void +btnMaxClicked(VBox, RangedFunction): void +btnMinClicked(VBox, RangedFunction): void +btnCalcYClicked(VBox, RangedFunction, String): void +btnYInterceptClicked(VBox, RangedFunction): void +btnRootsClicked(VBox, RangedFunction): void +btnDerivativeClicked(VBox, RangedFunction, String): void +btnAreaUnderClicked(VBox, RangedFunction): void +btnAreaBetweenClicked(VBox, RangedFunction): void +btnVolumeClicked(VBox, RangedFunction): void +btnIntersectionClicked(VBox, RangedFunction, RangedFunction): void +btnStudentsClicked(String): void +btnDoneStudentsClicked(): void +btnDeleteStudentClicked(String, String, String): void +btnAddStudentsClicked(String): void +btnDoneAddingStudentClicked(String, String, String): void +btnCancelAddingStudentClicked(): void +btnDeleteClassClicked(String): void +btnAddClicked(): void +btnDoneClassesClicked(): void +btnDoneAddingClicked(String): void +throwError(String, Stage): void +saveClasses(): void

Figure 10.2: Student

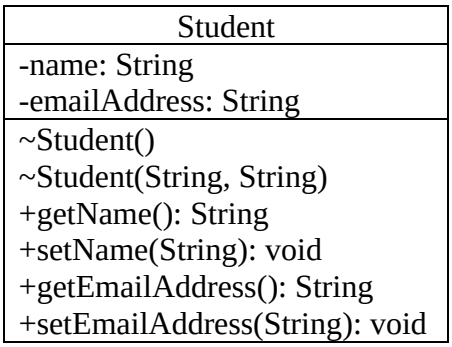


Figure 10.3: Class

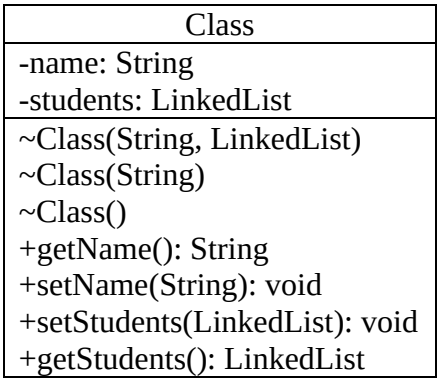


Figure 10.4: Validate

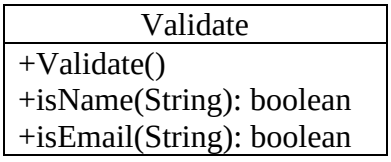


Figure 10.5: RangedFunction

RangedFunction
-function: Function -lower: double -upper: double
~RangedFunction(RangedFunction) ~RangedFunction(Function, double, double) ~RangedFuntion() +getFunction(): Function +getLower(): double +getUpper(): upper +setLower(double): void +setFunction(Function): void +setUpper(double): void

11. Testing strategy

Testing against the criteria of success will be done according to the following table.

Figure 11.1a: Testing strategy

Test to be performed	Criteria satisfied	Desired response
Delete an existing class	15	Class should be deleted
Add a new class	15	New class should be created
Edit the newly created class to contain one student	15	Student should be added to class
Input invalid student name “IA34”	17	Appropriate error should be thrown
Input invalid student email “ib cs”	17	Error should be thrown
Input invalid function “xyz”	17	Appropriate error should be thrown
Input valid function y=|x|*cos x in invalid range “zero” to “%%”	17	Appropriate error should be thrown
Graph y = |x|*cos x in the interval [0,30]	1	Function should be drawn
Find local minima for this funtion	2	Local minima should be calculated
Find local maxima for this function	3	Local maxima should be calculated

Figure 11.1b: Testing strategy (Continue)

Find y-coordinate when x=6	4	The result should be 5.761*
Find roots	5	Roots should be calculated, at the points where f(x) = 0
Find y-intercept	6	The result should be 0
Find derivative when x=3	7	The result should be -1.413*
Calculate area under graph	8	The result should be -30.487*
Calculate area between graph and x-axis	9	The result should be 283.673*
Calculate the volume of revolution	10	The result should be 13899.388*
Save data in pdf	13	Graph and data should be saved in pdf
Graph y=sin x in the interval [2,10] together with the previous function	11	Both graphs should be graphed
Find points of intersection of these graphs	12	The results should be (4.493, -0.976) , (7.725, 0.992) *
Save data in pdf	13	Graph and data should be saved in pdf
Send pdf to students of newly created class	14	Pdf should be sent via email to the class ‘students
Send pdf to students of different class	14	Pdf should be sent via email to the class’ student
Turn off internet connectivity and try to send pdf to different class	14	Program should not crash
Run all tests on smartboard	16	Everything should look good

*Calculated by GDC with 3 decimals

Word Count: 305