**Design**

Contents

[1. Use cases 2](#_Toc35455876)

[2. First visualizations 3](#_Toc35455877)

[3. Final visualizations 8](#_Toc35455878)

[4. Data types 13](#_Toc35455879)

[5. Flowcharts 14](#_Toc35455880)

[6. System flowcharts 20](#_Toc35455881)

[7. Pseudocode 22](#_Toc35455882)

[8. Class responsibilities 26](#_Toc35455883)

[9. Class connections 27](#_Toc35455884)

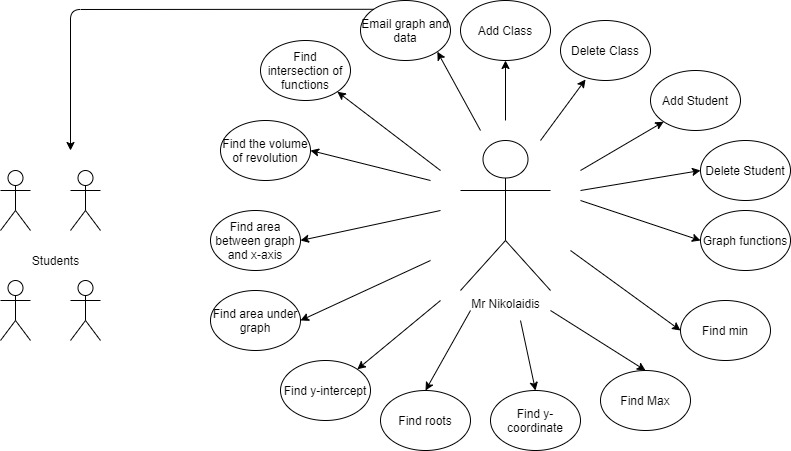
[10. UML diagrams 27](#_Toc35455885)

[11. Testing strategy 30](#_Toc35455886)

# Use cases

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

**Figure 1.1: Use cases**

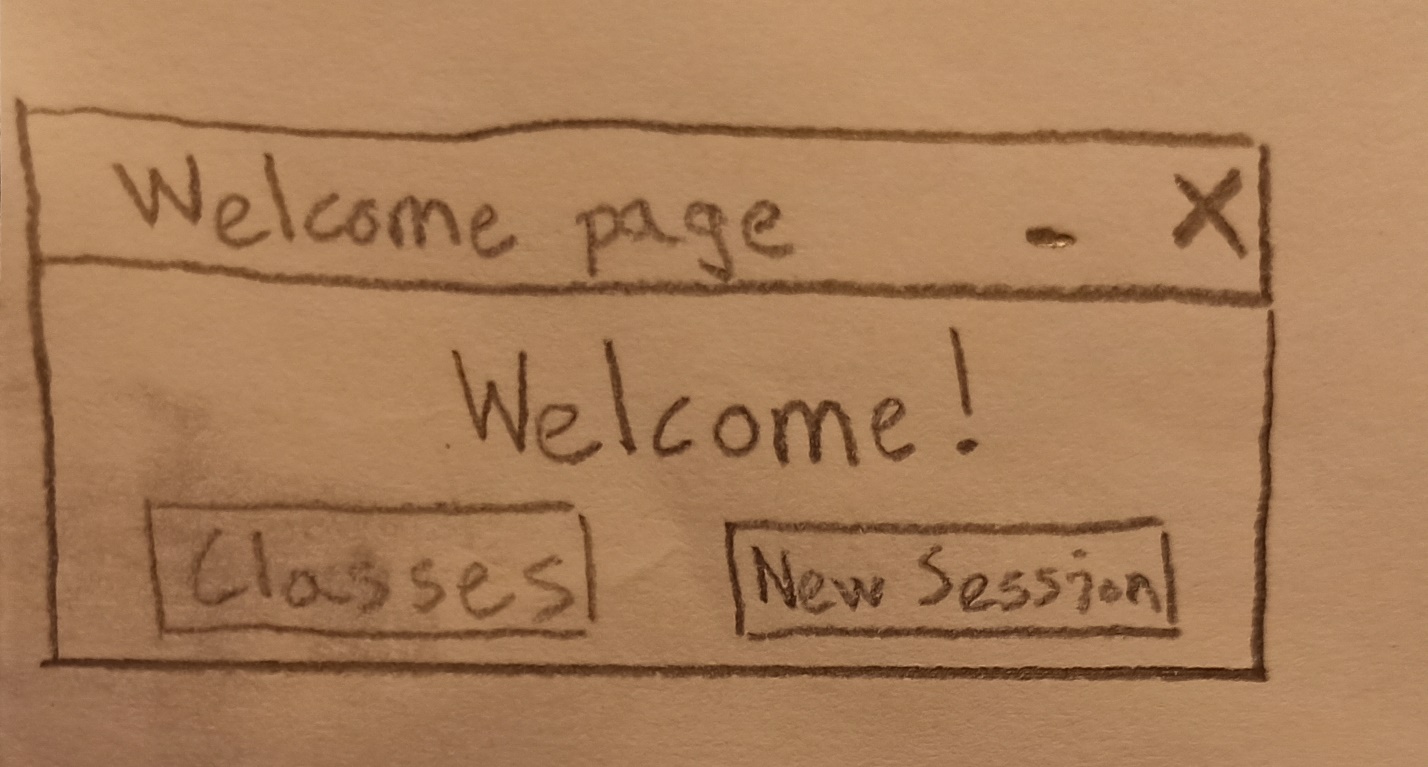


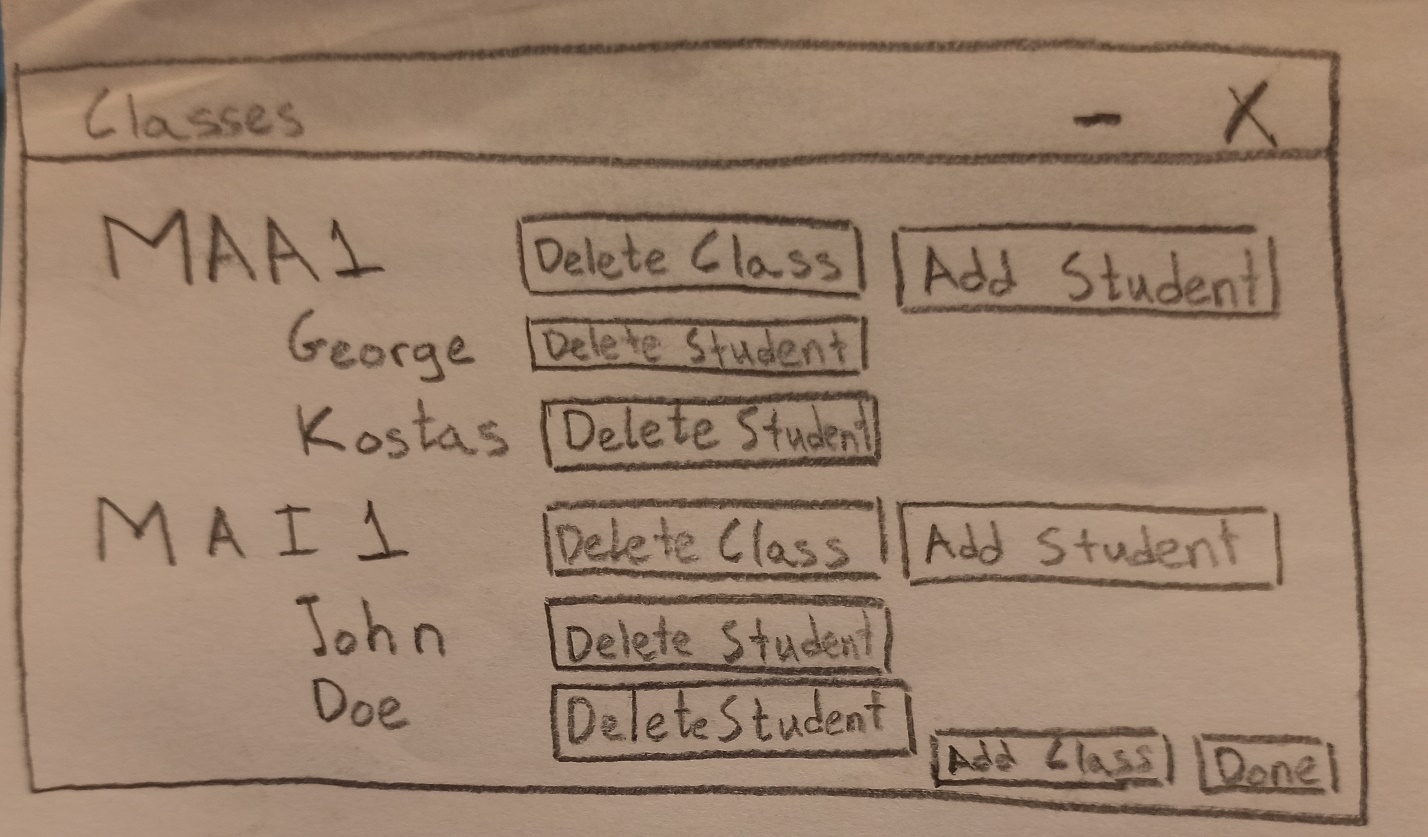
Mr. Christos

# First visualizations

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

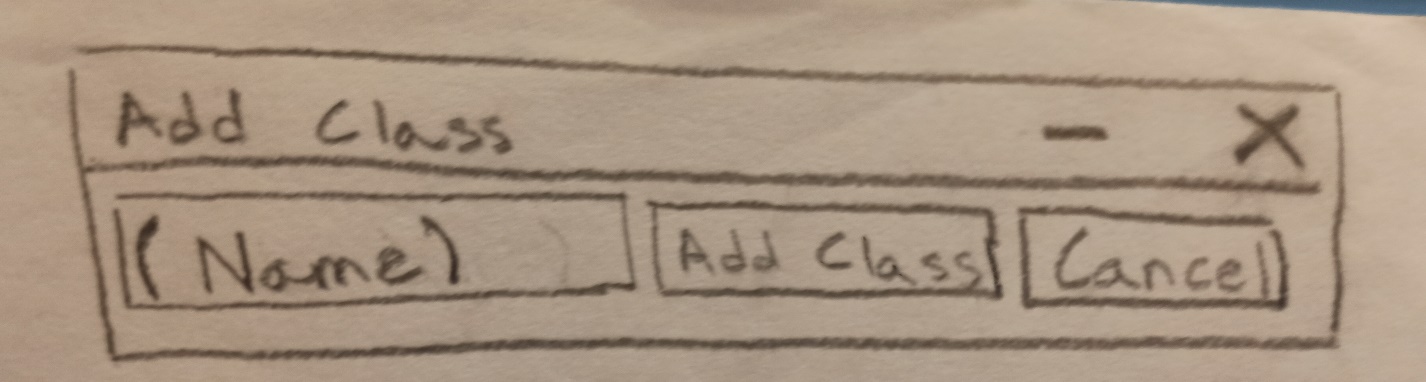
**Figure 2.1: “Welcome page” window**



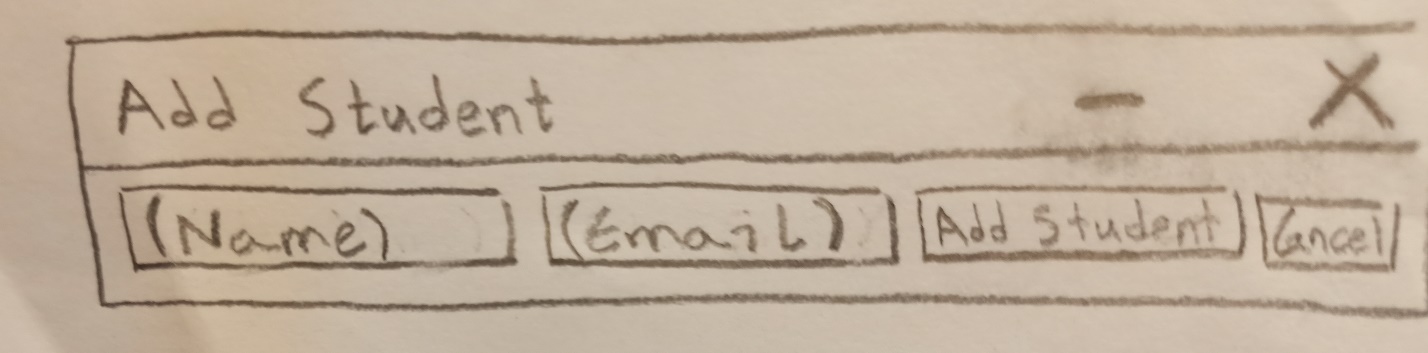


**Figure 2.2: “Classes” window**

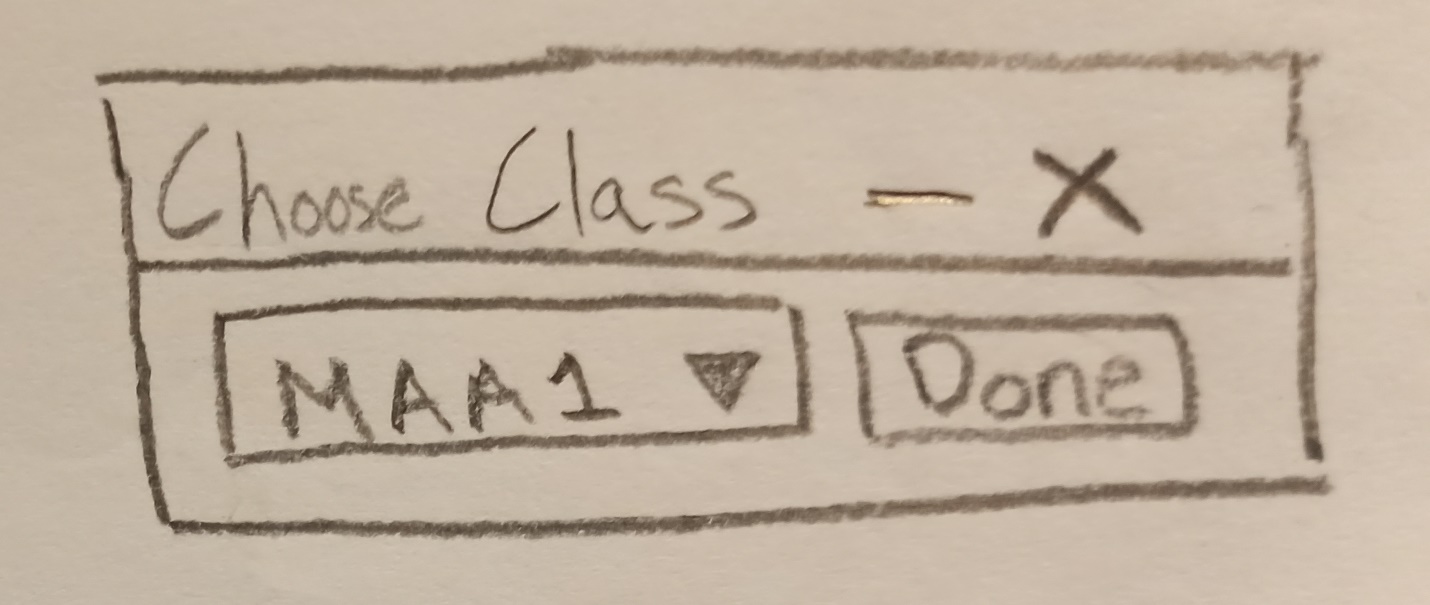
**Figure 2.3: “Add Class” window**



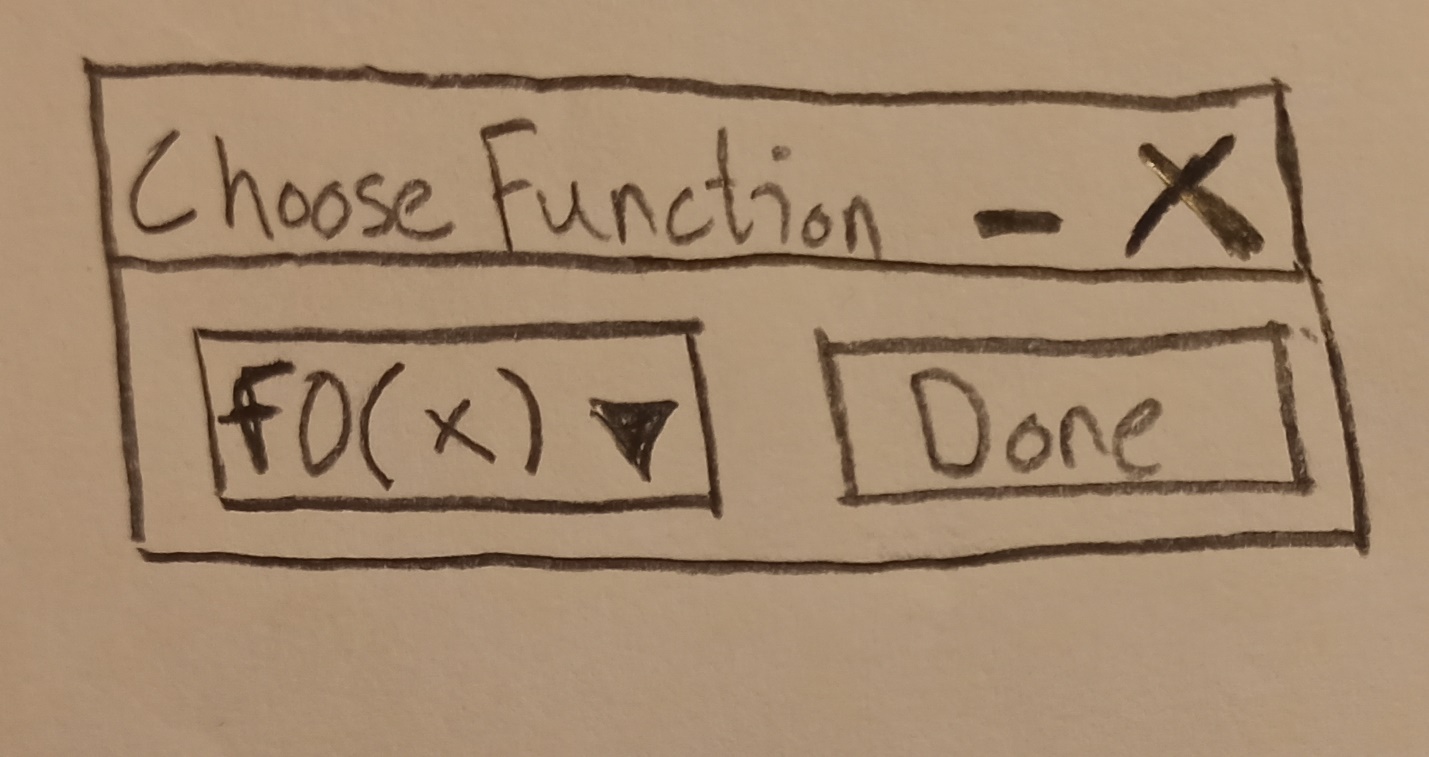
**Figure 2.4: “Add Student” window**



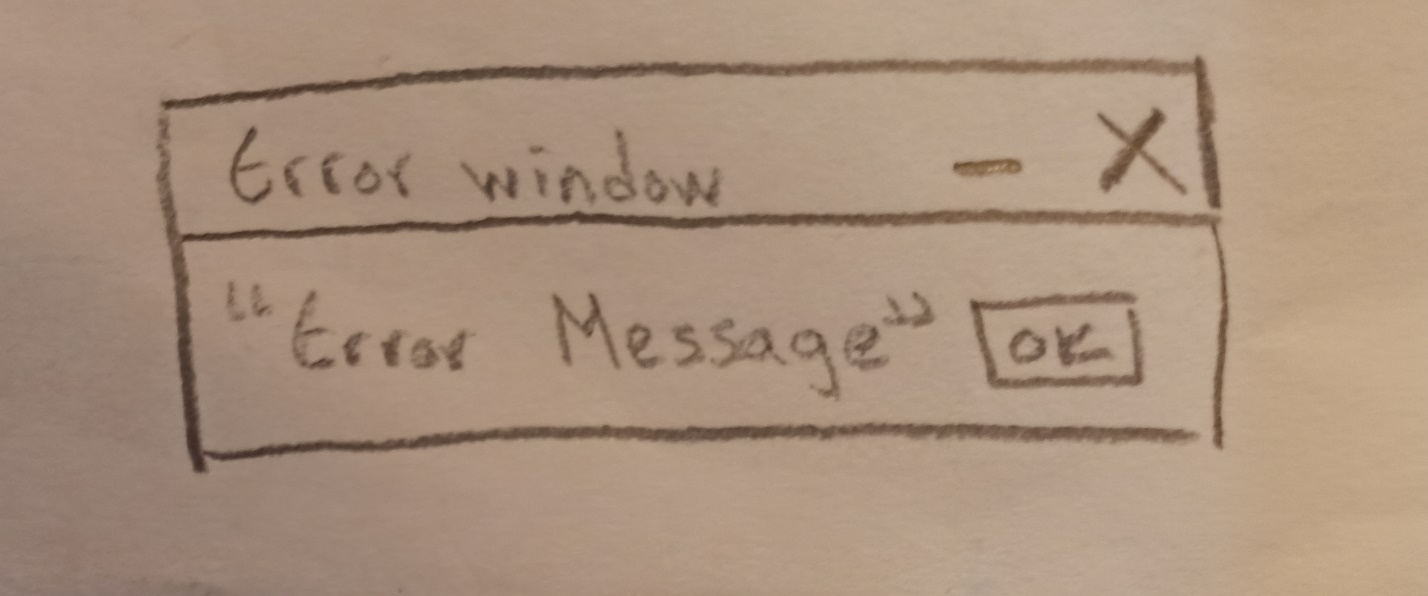
**Figure 2.5: “Choose Class” window**

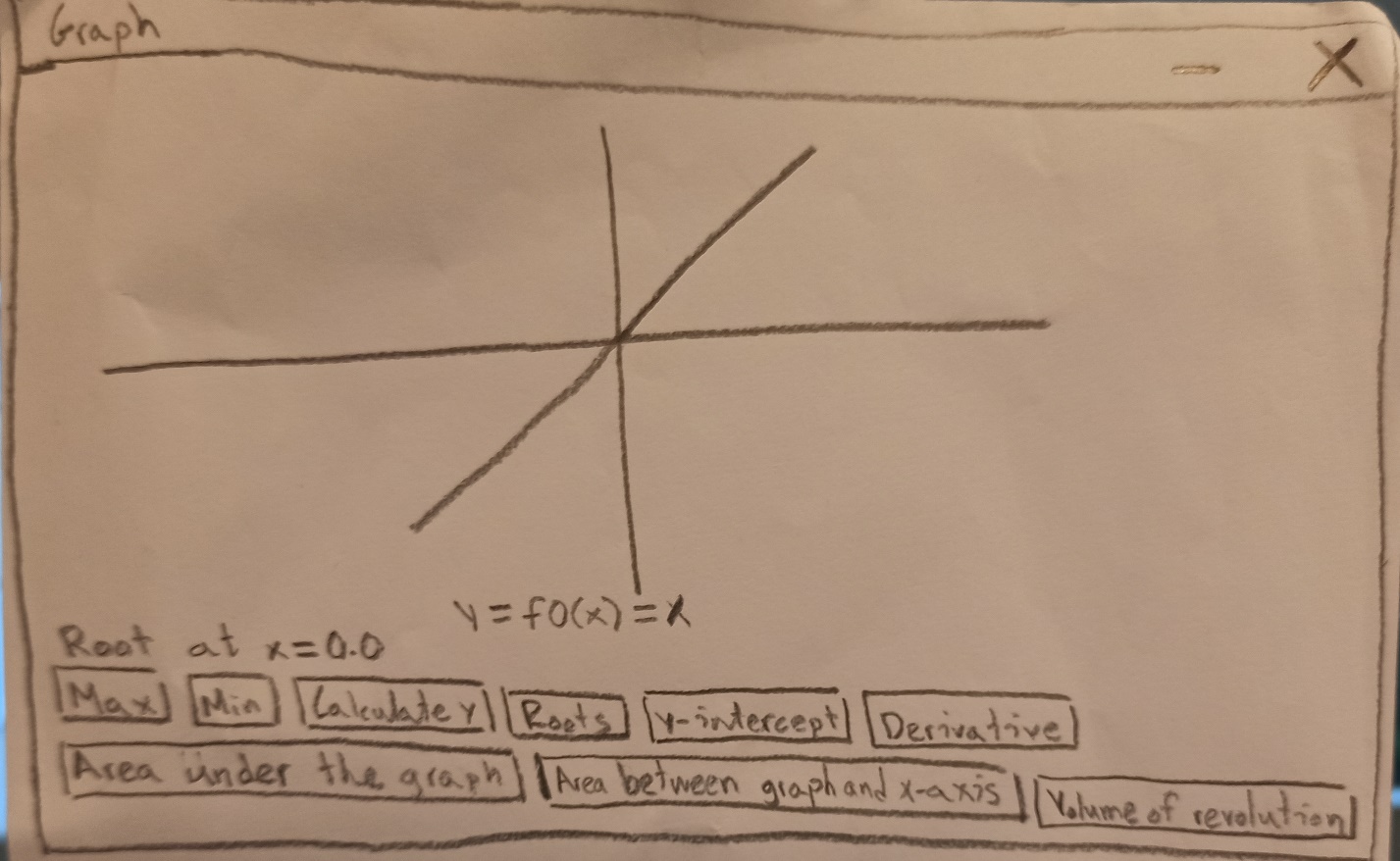


**Figure 2.6: “Functions” window**



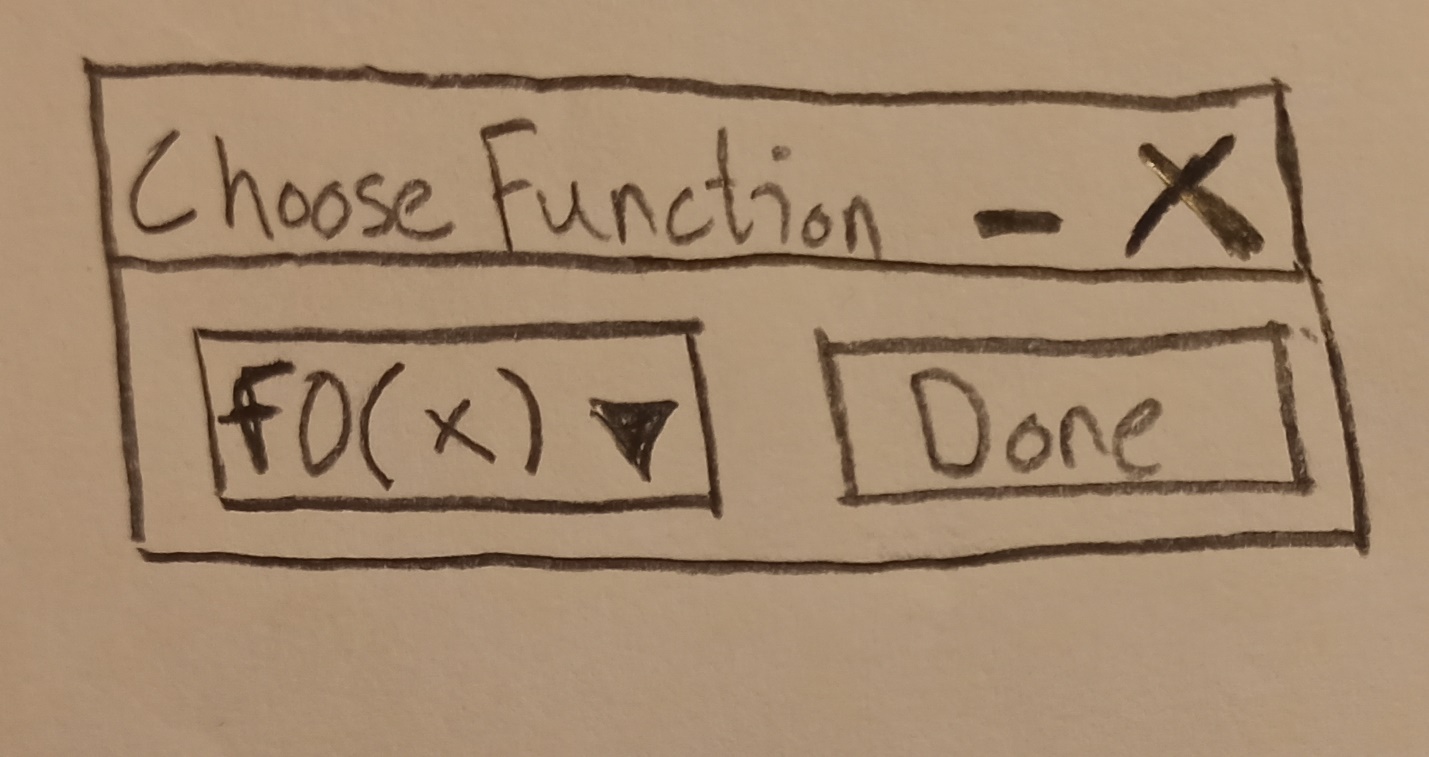
**Figure 2.7: “Error window”**





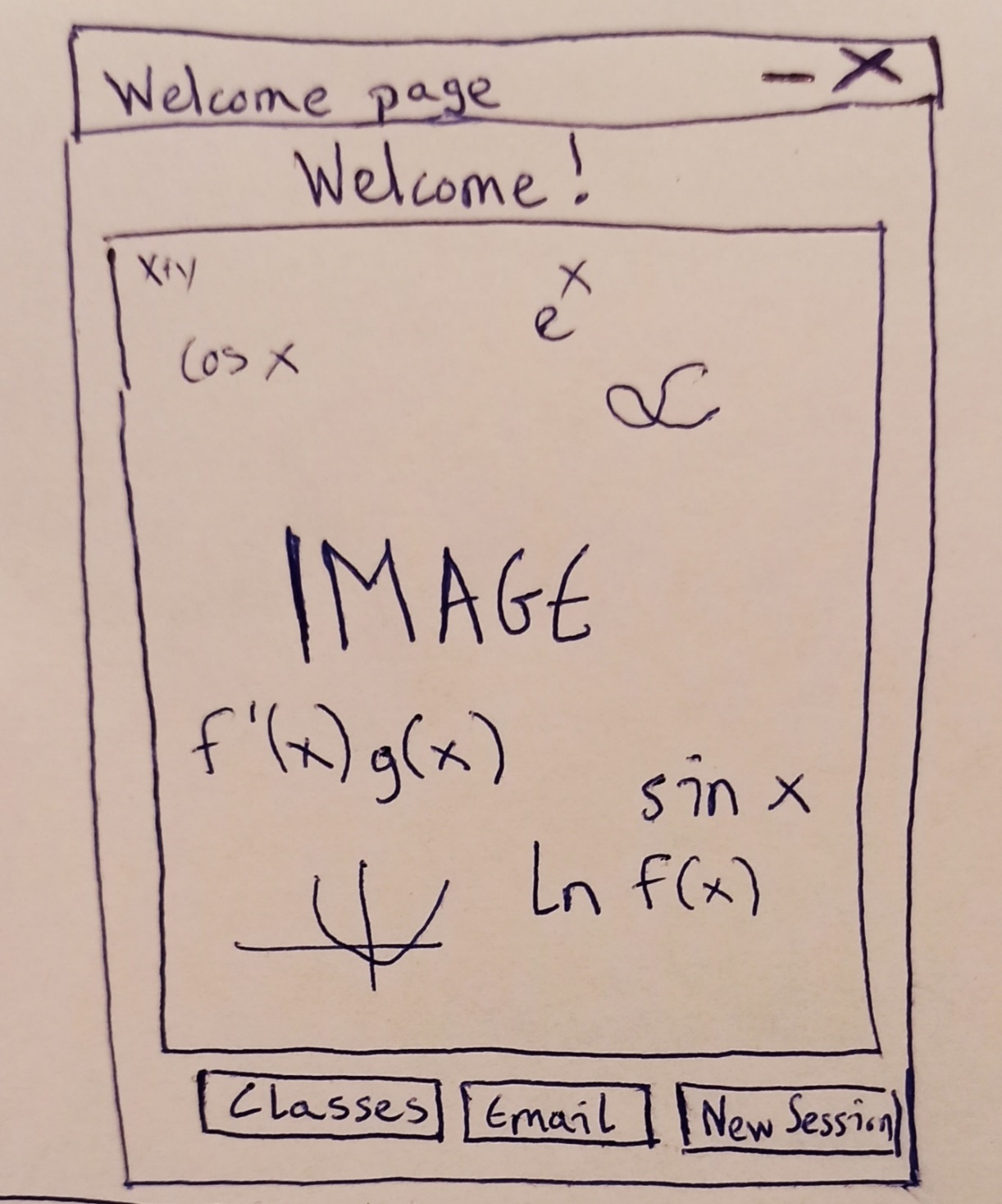
**Figure 2.8: “Graph” window**

**Figure 2.9: “Choose Function” window**



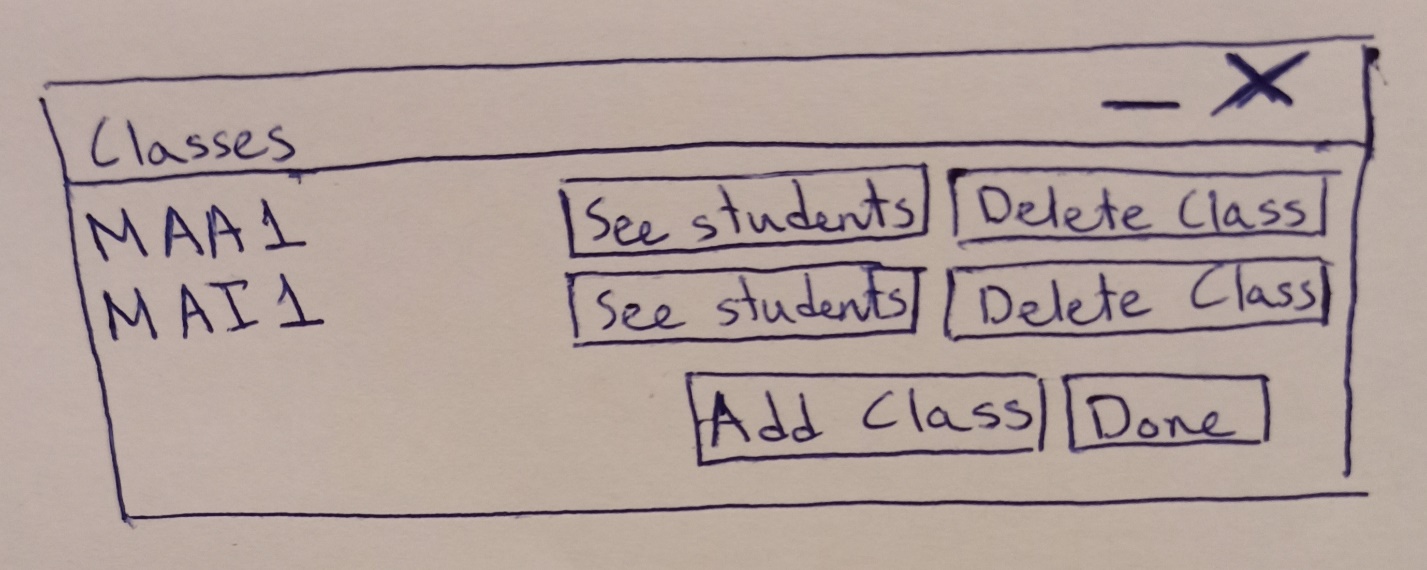
# Final visualizations

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

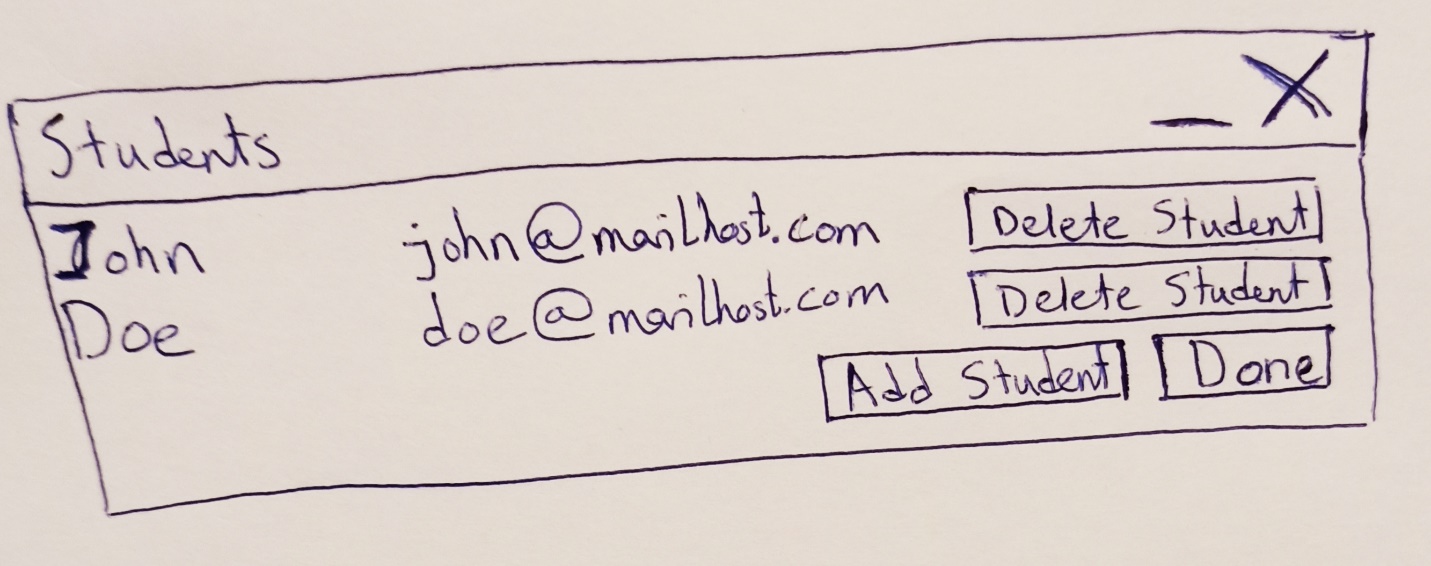


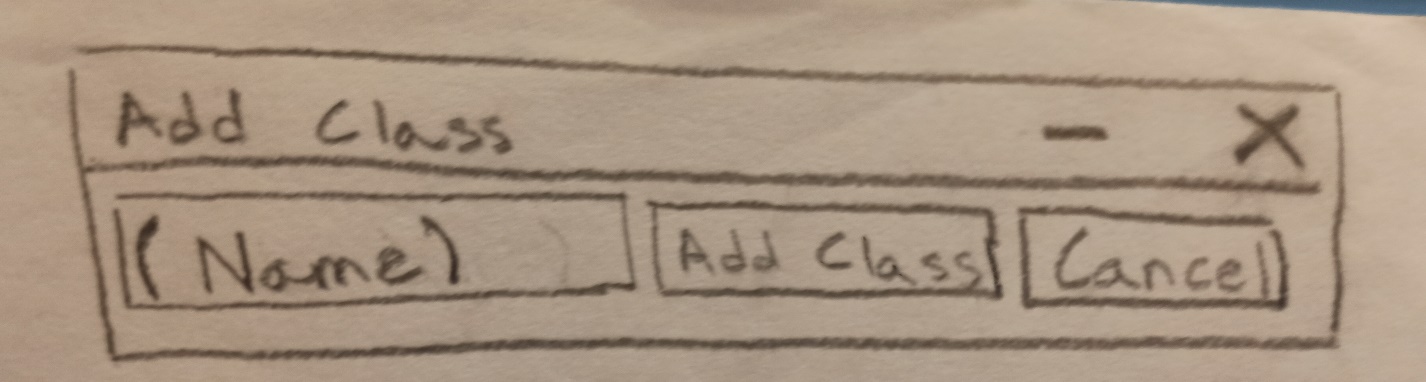
**Figure 3.1: “Welcome page” window**

**Figure 3.2: “Classes” window**



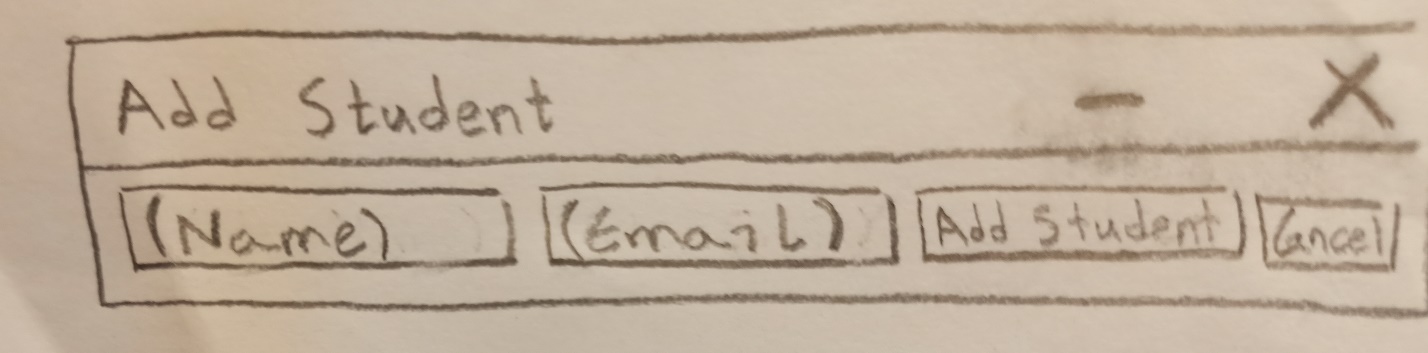
**Figure 3.3: “Students” window**



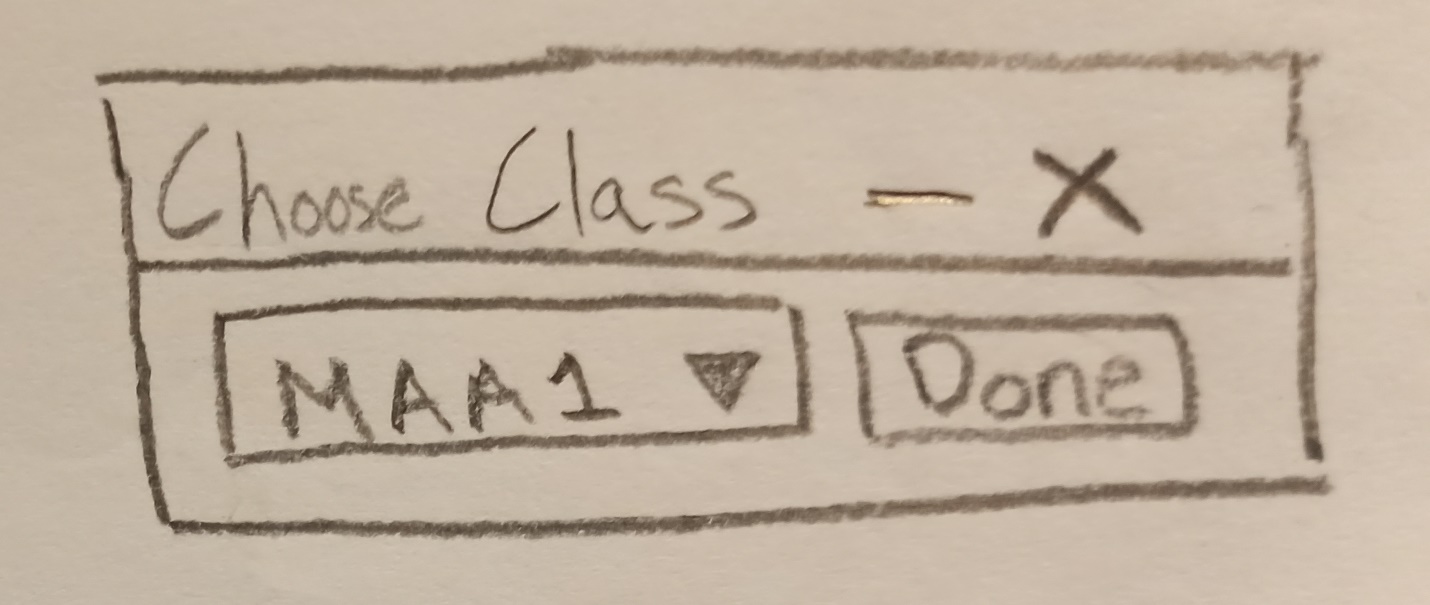


**Figure 3.4: “Add Class” window**

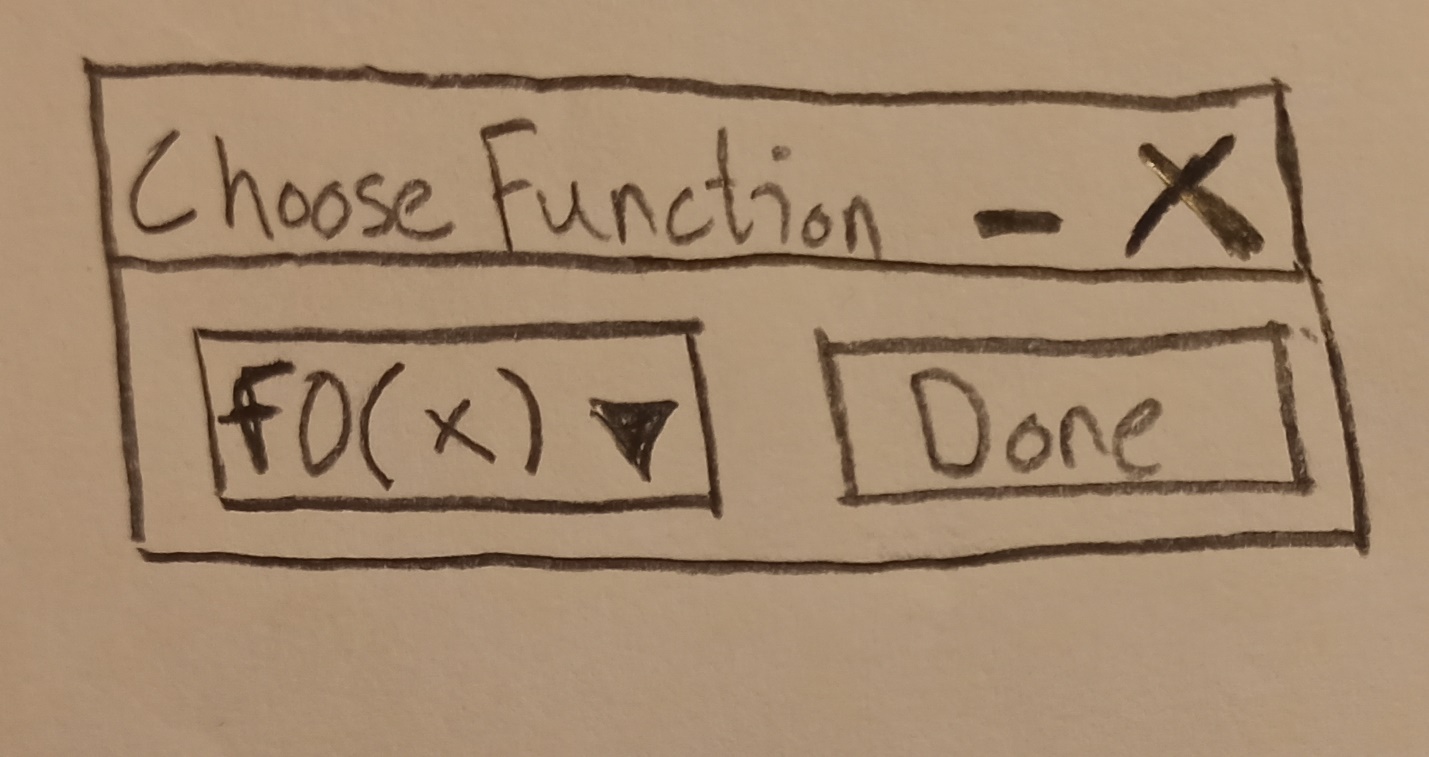
**Figure 3.5: “Add Student” window**



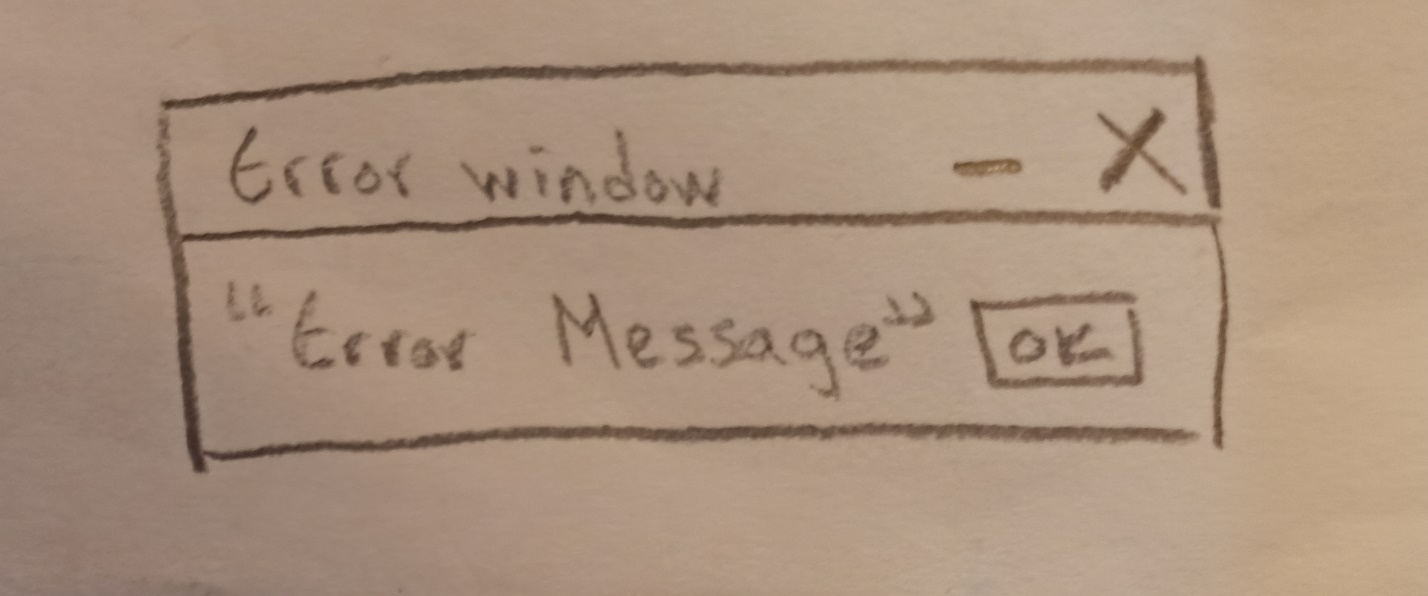
**Figure 3.6: “Choose Class” window**

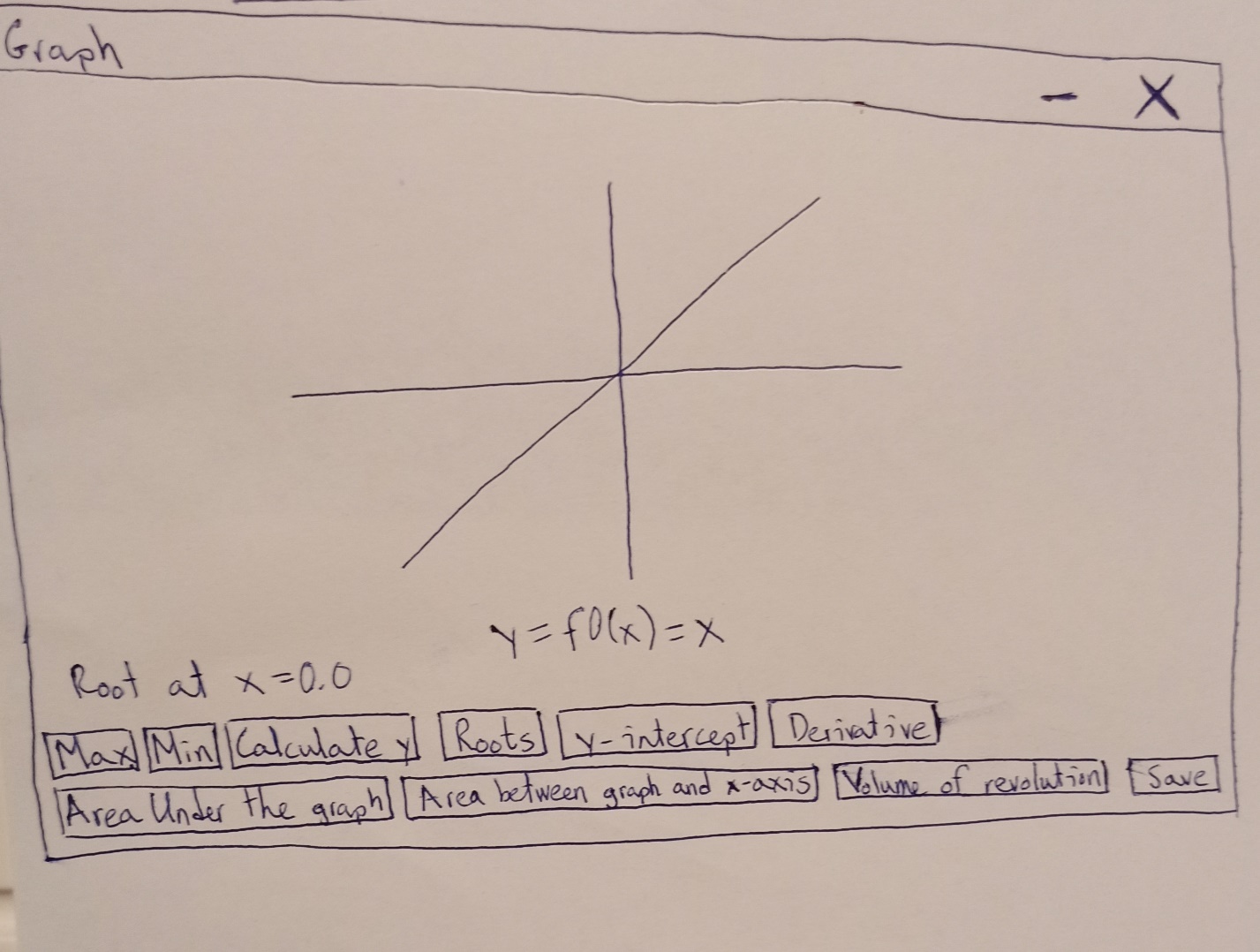


**Figure 3.7: “Functions” window**

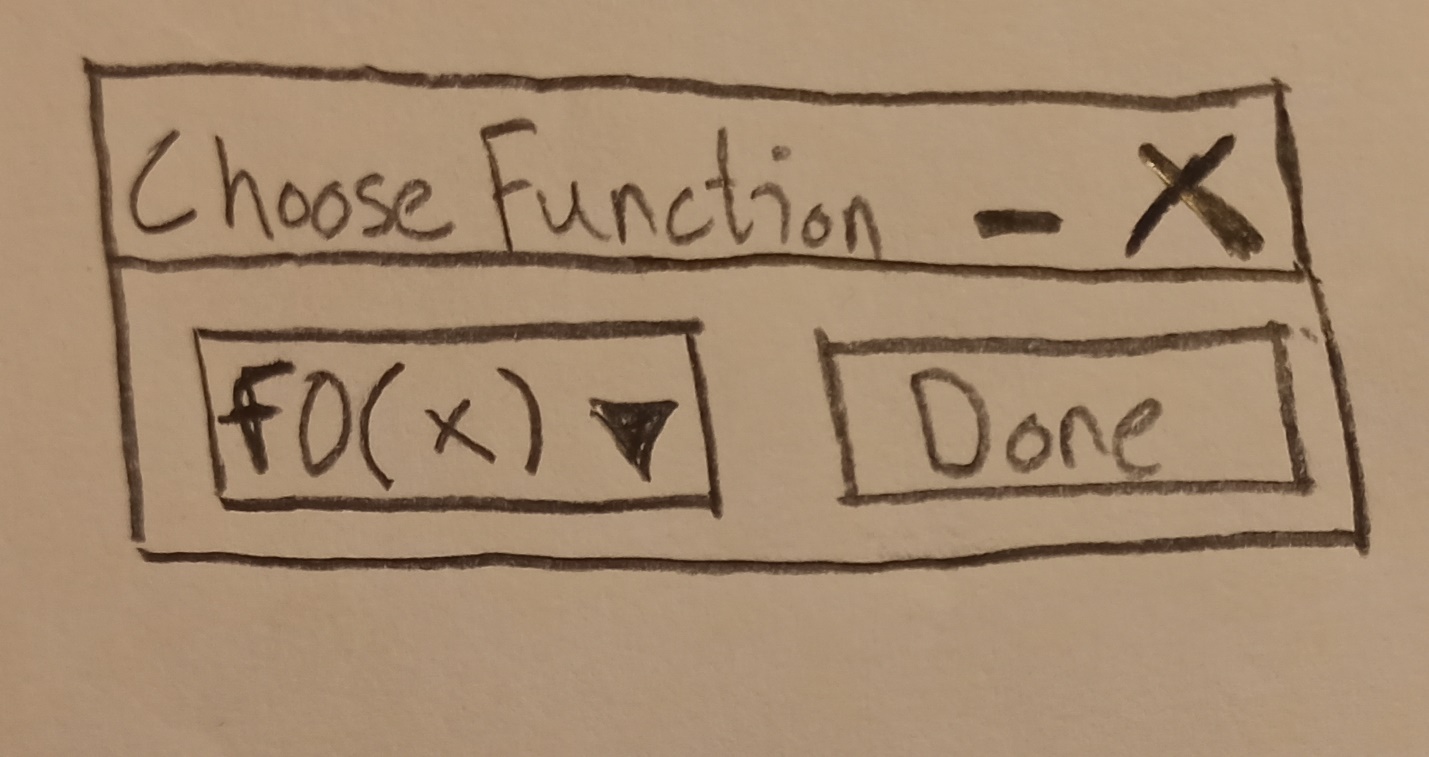


**Figure 3.8: “Error window”**





**Figure 3.9: “Graph” window**



**Figure 3.10: “Choose Function” window**

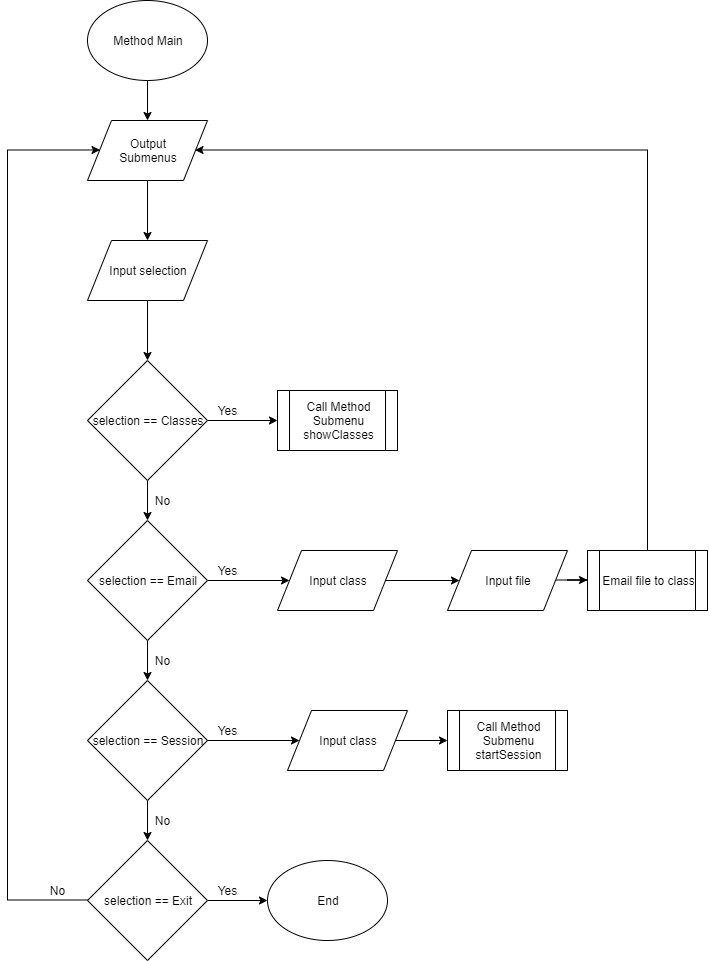
# Data types

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

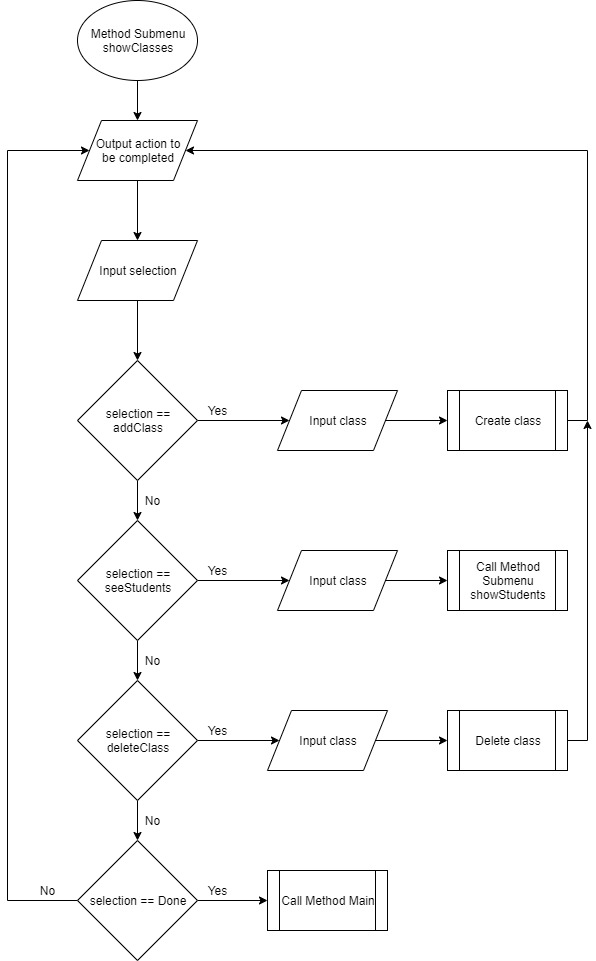
**Figure 4.1: Data types**

|  |  |
| --- | --- |
| **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 |

# Flowcharts

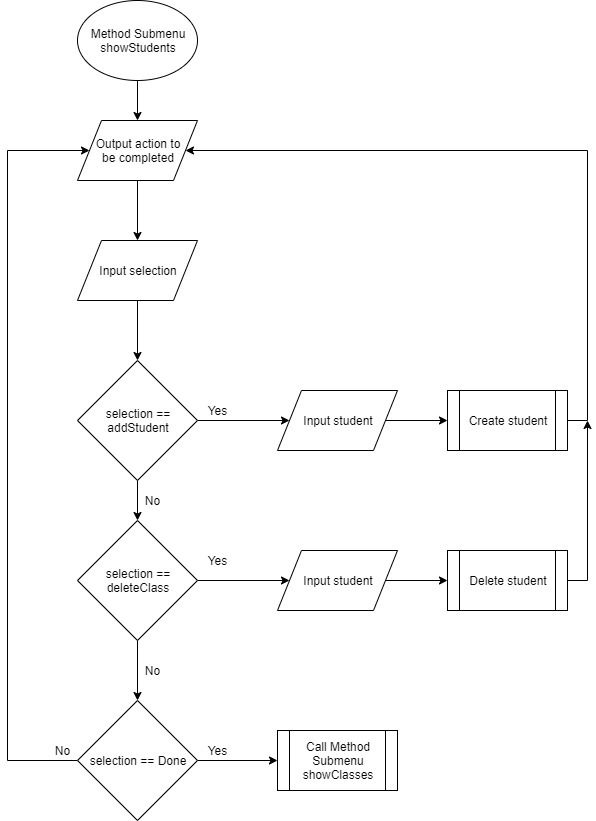
The flowcharts show the menus and different options that the user will have in every case.

**Figure 5.1: Method Main**

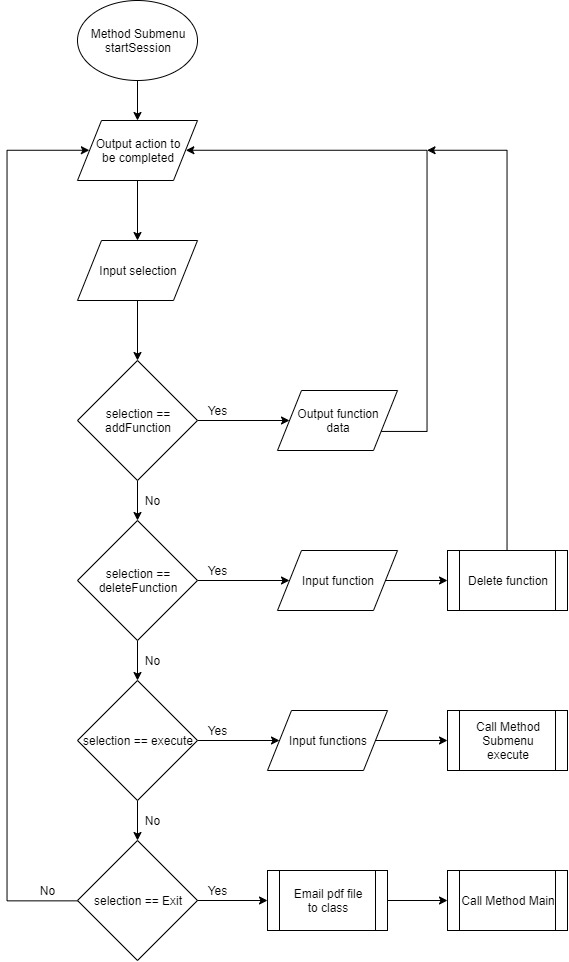


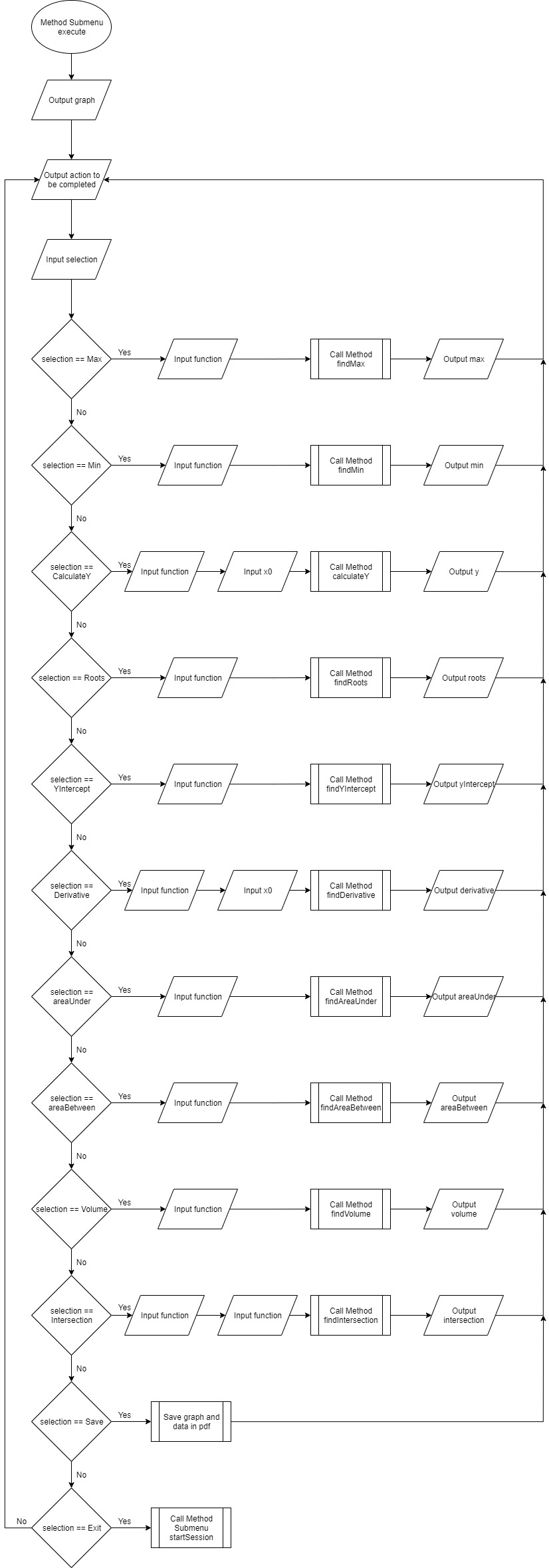
**Figure 5.2: Method Submenu showClasses**

**Figure 5.3: Method Submenu showStudents**

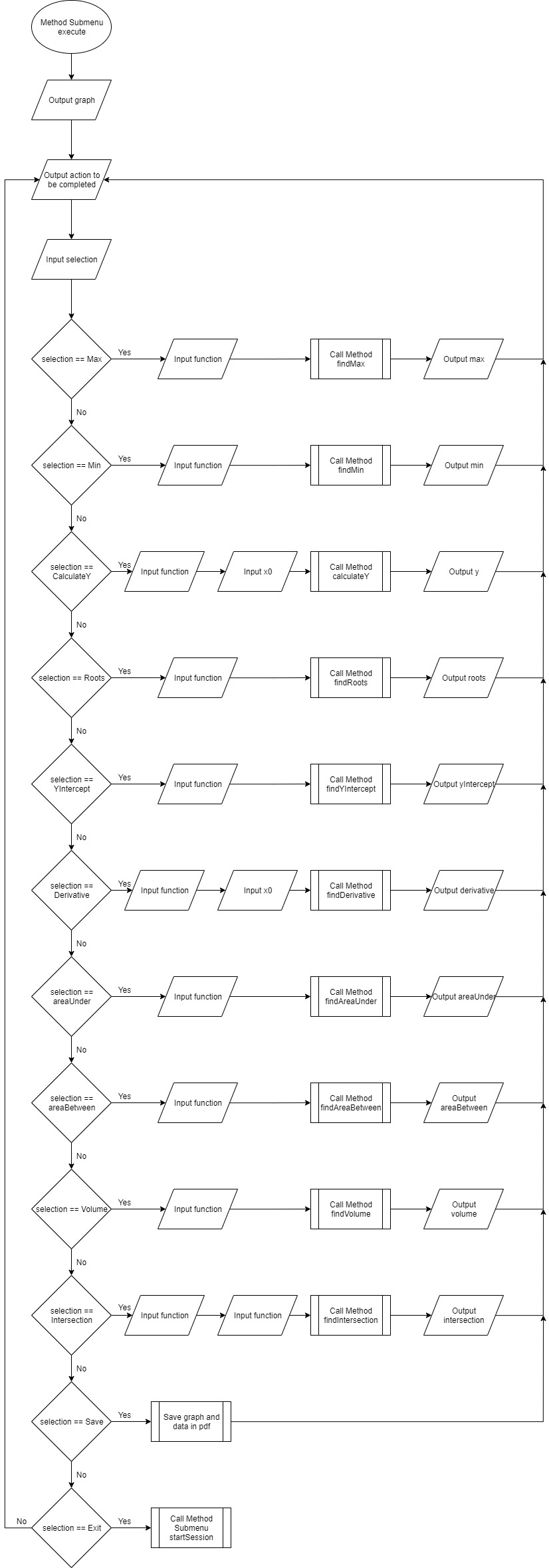


**Figure 5.4: Method Submenu startSession**





**Figure 5.5a: Method Submenu execute**

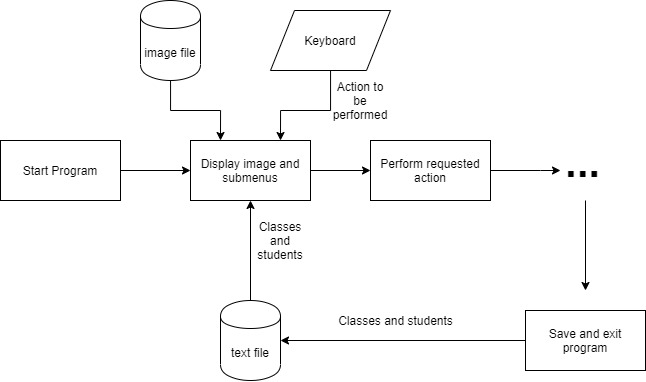


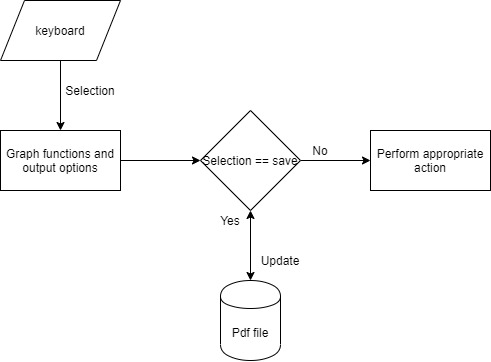
**Figure 5.5b: Method Submenu execute (Continued)**

# System flowcharts

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

**Figure 6.1: Main**





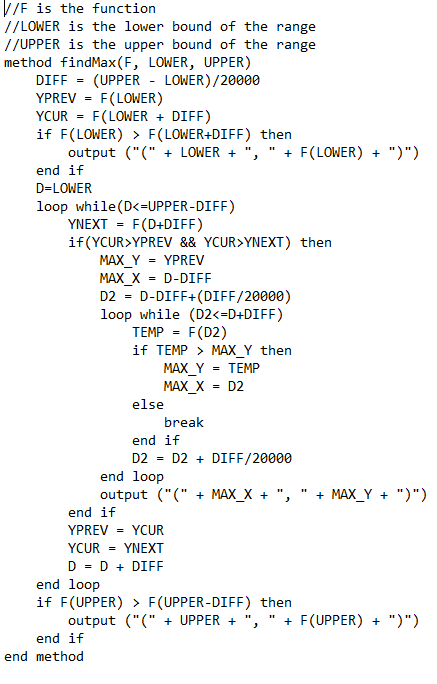
**Figure 6.2: Graph**

# Pseudocode

The pseudocode of some complex algorithms is presented below.

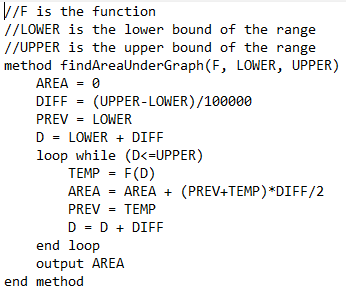
* findMax precondition: a given plotted function within a range
* findMax postcondition: the coordinates local maxima are printed

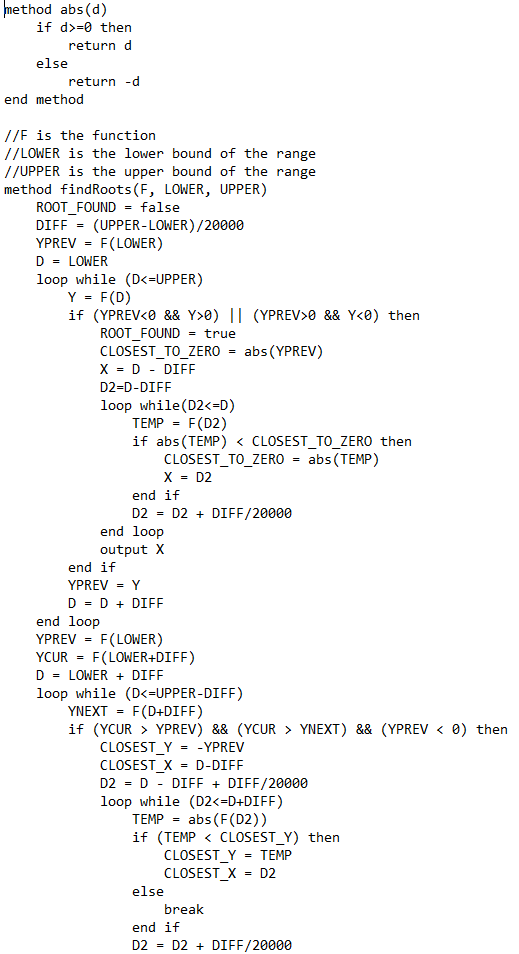
**Figure 7.1: findMax**



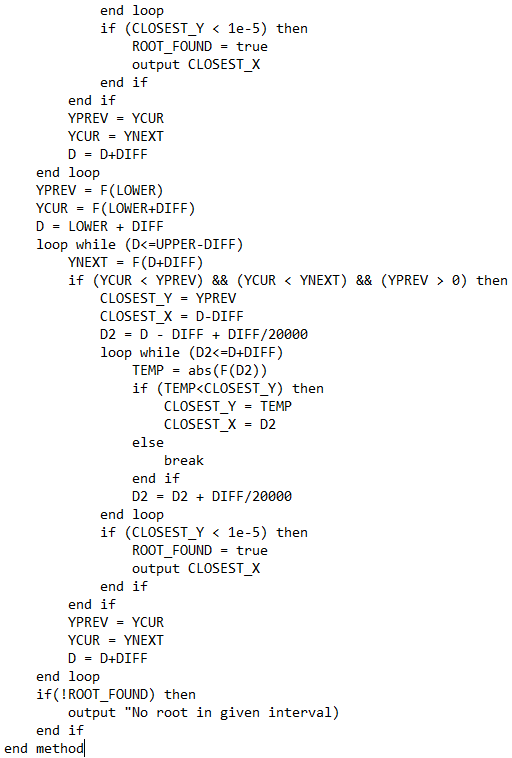
* findAreaUnderGraph precondition: a given plotted function within a range
* findAreaUnderGraph postcondition: the area under the graph of the function is printed
* findRoots precondition: a given plotted function within a range
* findRoots postcondition: the values of x for which f(x)=0 are printed

**Figure 7.2: findAreaUnderGraph**





**Figure 7.3a: findRoots**



**Figure 7.3b: findRoots (Continued)**

# Class responsibilities

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

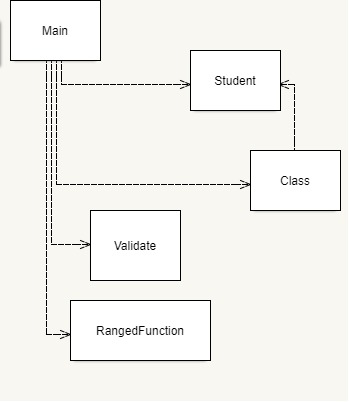
**Figure 8.1: Class responsibilities**

|  |  |
| --- | --- |
| 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 |

# Class connections

The figure shows the relationship between the classes.

**Figure 9.1: Class connections**



# 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  -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 10.1b: Main (Continued)**

**Figure 10.2: Student**

|  |
| --- |
| Student |
| -name: String  -emailAddress: String |
| ~Student()  ~Student(String, String)  +getName(): String  +setName(String): void  +getEmailAddress(): String  +setEmailAddress(String): void |

|  |
| --- |
| Class |
| -name: String  -students: LinkedList |
| ~Class(String, LinkedList)  ~Class(String)  ~Class()  +getName(): String  +setName(String): void  +setStudents(LinkedList): void  +getStudents(): LinkedList |

**Figure 10.3: Class**

**Figure 10.4: Validate**

|  |
| --- |
| Validate |
| +Validate()  +isName(String): boolean  +isEmail(String): boolean |

**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 |

# 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 |
| 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

**Figure 11.1b: Testing strategy (Continue)**

Word Count: 305

1. see Appendix B [↑](#footnote-ref-1)