TCP1101 ASSIGNMENT PROJECT REPORT
by GROUP 7

TEAM LEADER

Srihari Naidu A/L Venkatash	+6010-667 7569	1201100182@ student.mmu.edu.my

TEAM MEMBERS

Mohammed EmadEldin Ahmed Taha Elmogy	+20 1501081422	1191102751@ student.mmu.edu.my
Najmuddin Bin Thijs Hovenkamp	+6019-820 9317	1201100495@ student.mmu.edu.my
Muhammad Aiman Haziq Bin Fairoshawal	+6018-950 4018	1201101381@ student.mmu.edu.my

TABLE OF CONTENTS

INTRODUCTION

The Data Analysis Tool (BDAP) is a software programme that allows users to automatically evaluate data files that they give. Users should be able to evaluate data and create reports using the software. Until the user quits the system, the application will retain track (log) of the user and their actions.

The program that we are creating is used to find and calculate statistics such as Mean, Variance and even plotting a histogram based on data gathered from a data file.The program was mainly created to simply help ease the calculation as doing it manually can take more time and energy,which could lead to miscalculations or errors in the data.

The tasks have been split across 4 teammates to give everybody in the group the same amount of work division possible which can be viewed in the previous page for how we split the work on our code, and for the preot everyone is taking part in it but we split some tasks to make it quicker in progress like someone doing the research and other doing flowcharts and arranging the screenshots from the menus, etc. As a side note everyone created their own flow charts based on the work division they have been assigned to do.

WORK DIVISION

F1. User Registration and Requirements		
F1.1	Create user account	Hari, Emad
F1.2	Login to a user account	Hari, Emad
F1.3	Logout from a user account	Hari
F1.4	Delete user account	Aiman, Najmuddin
F1.5	Change user password	Aiman, Najmuddin

F2. Data File Reader		
F2.1	Load data file	Hari, Najmuddin
F2.2	Save as	Hari, Najmuddin
F2.3	Save report	Emad, Aiman, Hari, Najmuddin
F2.4	Save report HTML	Emad, Aiman, Hari, Najmuddin
F2.5	Log	Emad

F3. Statistical Computation		
F3.1	Find Minimum	Najmuddin
F3.2	Find Maximum	Najmuddin
F3.3	Median	Emad
F3.4	Mean	Emad
F3.5	Variance	Aiman
F3.6	Standard Deviation	Aiman

F3.7	Correlation between any selected 2 columns	Hari
F3.8	Distinct data members	Hari
F3.9	Plot A histogram (text mode)	Hari

F4. Menu System		
F4.1	Main Page	Hari
F4.2	Statistical Analysis Menu	Emad
F4.3	Report Menu	Emad, Aiman, Hari, Najmuddin
F4.4	HTML generated Report	Emad, Aiman, Hari, Najmuddin
F4.5	Error Checking	Hari

FUNCTIONAL REQUIREMENTS

F1. User Registration and Requirements		
F1.1	Create user account	Only the admin can create the user accounts. The data needed to create an account would be the user account, password, and user type as well. After that all the data is stored in a separate database and a status of the user is also shown after creating the account.
F1.2	Login to a user account	Users can login using the account they have created by entering the username and password which is compared with the information stored in the database.
F1.3	Logout from a user account	After the user has logged in into his account, they can also logout from the system and go to the main menu again where the user can login again.
F1.4	Delete user account	Only the admins can delete the user accounts by typing the username which is compared to the stored database if it matches the account gets deleted successfully.
F1.5	Change user password	Admin and the user can change his/her password after they login to their accounts and choose the change user password option.

F2. Data File Reader		
F2.1	Load data file	The program loads the data from the file into a proper data structure array.
F2.2	Save as	The program saves the data that has been done in the program as a new file in the file format the user prefers.
F2.3	Save report	The program saves the report in text file format.
F2.4	Save report HTML	The program saves the report in a single HTML format to be displayed using a web browser.
F2.5	Log	The program saves all the activities from when the user logs in and logs out, in a text file.

F3. Statistical Computation		
F3.1	Find Minimum	The program finds the minimum of a specific row or column.
F3.2	Find Maximum	The program finds the maximum of a specific row or column.
F3.3	Median	The program finds the Median of a specific row or column.
F3.4	Mean	The program finds the Mean of a specific row or column.
F3.5	Variance	The program finds the Variance of a specific row or column.
F3.6	Standard Deviation	The program finds the Standard Deviation of a specific row or column.
F3.7	Correlation between any selected 2 columns	The program computes Pearson's correlation between any two selected columns showing the table of computation of the correlation (correlation measures the strength of association between two quantitative variables.
F3.8	Distinct data members	The program displays the distinct numbers with no repetition and a count of the occurrences of each frequency in a tabular form.
F3.9	Plot A histogram (text mode)	The program creates a histogram based on a specific column that is chosen by the user.

F4. Menu System		
F4.1	Main Page	The program displays the user's name, program title and options that are available for the user.
F4.2	Statistical Analysis Menu	The program displays The user options for the statistical menu and its functions.
F4.3	Report Menu	The program displays and generates reports based on the choosing of the users.
F4.4	HTML generated Report	The program generates the same report but in HTML form.
F4.5	Error Checking	The program displays errors if the user has given the wrong input when using the program.

MENU SYSTEM : DESIGN & MOTIVATION

A data analysis program is ought to be complex, but we had to keep the complexity away from the design and interface. B.D.A.P is a console-based program which requires us to construct our own design and layout from scratch. We aimed to keep the design clean & simple with a touch of ASCII Art for some designs; whilst being less confusing to operate.

Main Menu - Admin

This is the menu that's displayed to the Admin when he/she logs into the system. Admins are presented with admin-centric functionalities; Register Account and Manage Account - together with the basic Logout function.

Main Menu - B.D.A.P

This is the menu that's displayed to the User when he/she logs into the system. Users are instead presented with program-related functionalities; Statistical Analysis, Report, Log and Manage Account - together with the basic Logout function.

```
   ____    ____   ___    ____
  / _ )   /  _ \ / _ |  /  _ \
 / _  |  / / / // /\ |  / /_/ /
/ /_/ / / / / // /  \ |/ ____/
/_____(_)_____(_)_/  |_(_)_/

        Welcome, hari2
+-------------------------------+

 [1]  Statistical Analysis.
 [2]  Save Data File.
 [3]  Report.
 [4]  Manage Account.

 (l)  Logout.


+-------------------------------+

Choice [1234 l]
> ▮
```

Statistical Analysis Menu

This is the menu the User sees when he/she opts into the Statistical Analysis from the B.D.A.P Menu. They're given statistical functionalities; Minimum, Maximum, Median, Mean, Variance, Standard Deviation, Pearson's Correlation, Distinct Data Member, Histogram- together with the Back functionality if they wish to return to the B.D.A.P Menu.

```
  _____ _____ _____ _____     _____ _____ _____ _____ _____ _____
 /  ____/__  ___//  _/ |/  ___/    /  ___  /  ___  /  _   _|  ___|/  /  ___|  __   |
 \  \___/  / /  / / |  |/  |  __  /  /  |  /  /  |  /__   |/  |    \  __  /  \  \_  \
  \__   /  / /  / __|  |  _|  \/ /  __  \ |/  __  \/   | |  |      /  |  /  |  \__/ /
 /_____//_/ /_/ |_/(_) /_/   \__//_/  \_\//_/  \_\/____/_/|_____//___//____|_____/
```

```
    +----------------------------------------------------------------------+
    |
      [1]  Minimum.
      [2]  Maximum.
      [3]  Median.
      [4]  Mean.
      [5]  Variance.
      [6]  Standard Deviation.
      [7]  Pearson's Correlation.
      [8]  Distinct Data Member.
      [9]  Histogram.

      (b)  Back.


    +----------------------------------------------------------------------+

    Choice [12356789 b]
    > █
```

Reports Menu

This is the menu the User observes when he/she opts into the Report from the B.D.A.P Menu. They're presented with report-related functionalities; Save Report in plain text format or HTML format - together with the Back functionality if they wish to return to the B.D.A.P Menu.

STRUCTURED CHARTS
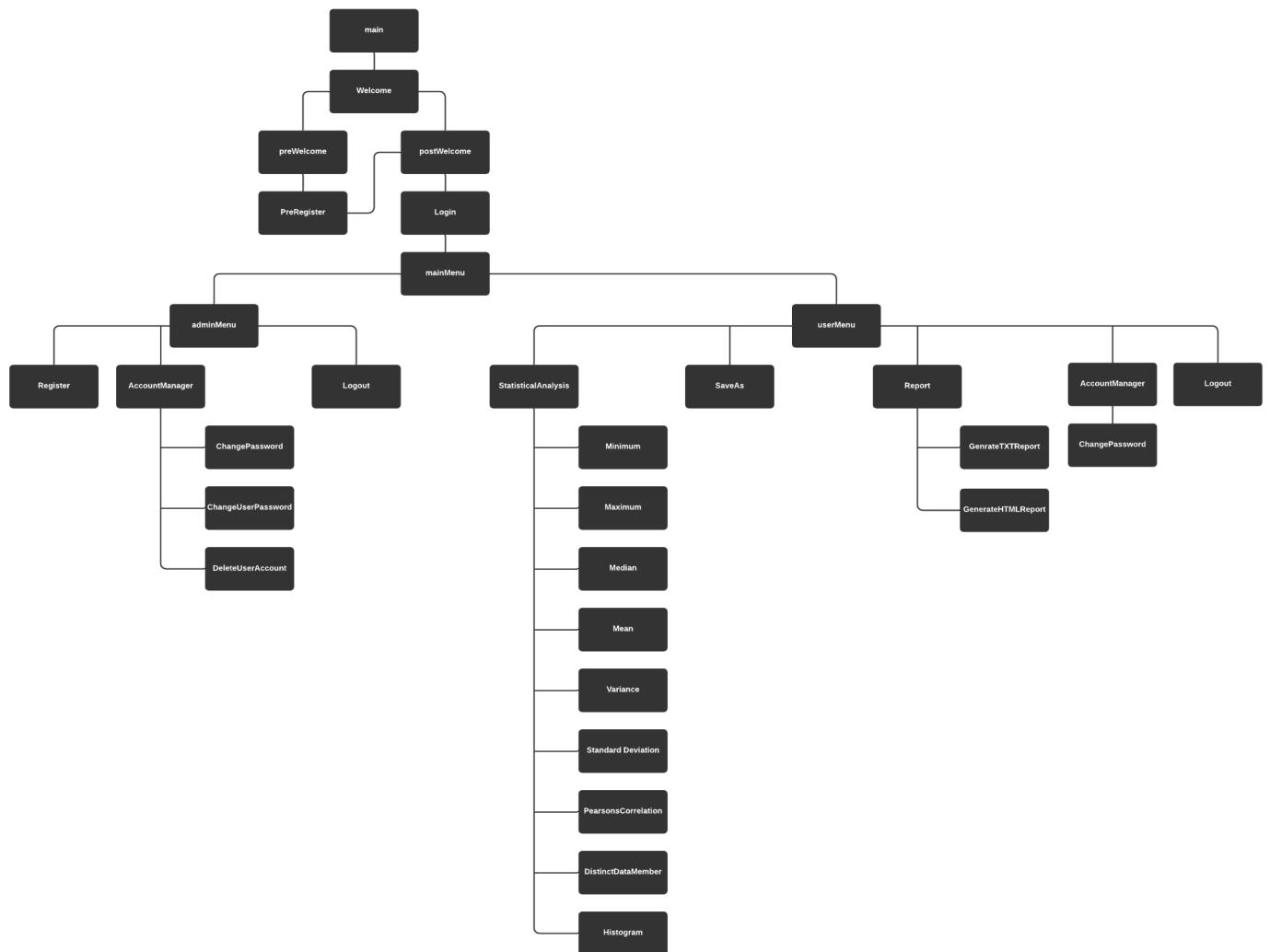
ALGORITHMS : FLOWCHARTS & PSEUDOCODE

F1 User Registration and Authentication

F1.1 Create user account
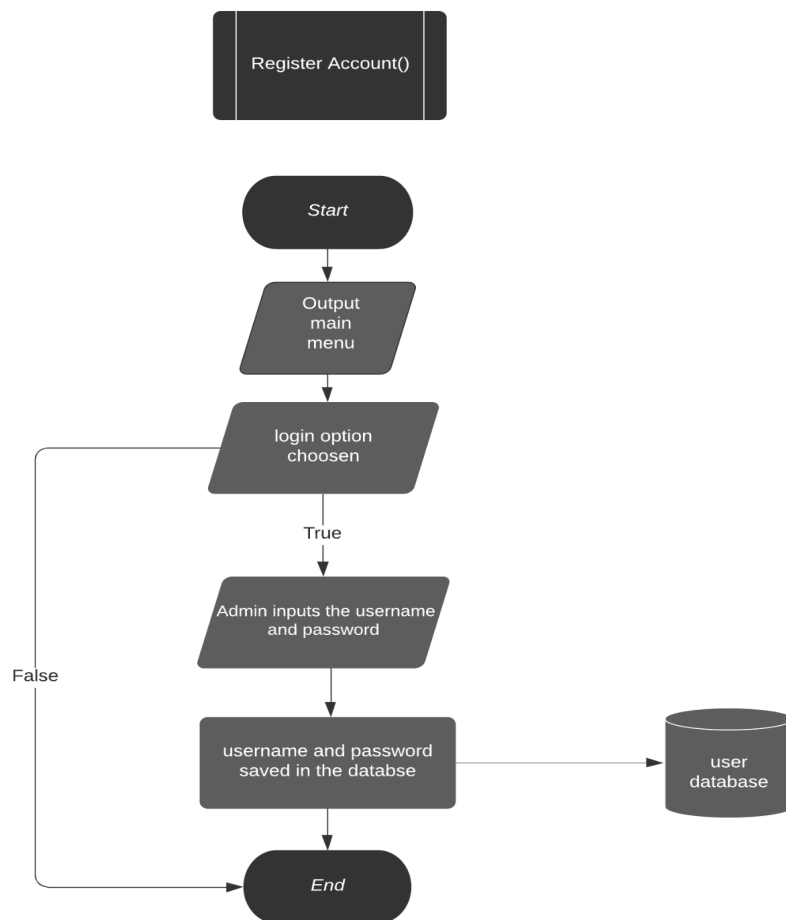
The create user account function can be only accessed by the administrator as the normal user cannot create his own account, the admin is presented with a login screen as he chooses that option, they have to enter their username and password to create an account which then saved into the database if there is not duplicates of the account saved the process should be successful.

F1.2 Login to a user account

```
┌─┬──────────────────┬─┐
│ │     Login()      │ │
└─┴──────────────────┴─┘

         ( Start )
            │
            ▼
   ╱─────────────────╲
  ╱ Admin/user inputs the username
 ╱      and password      ╲
╱─────────────────────────╲
            │
            ▼
   ┌──────────────────┐         ┌──────────┐
   │ username and password │───▶ │   user   │
   │ comapred with databse │     │ database │
   └──────────────────┘         └──────────┘
            │
            ▼
   ┌──────────────────┐
   │ data entered = same saved │
   │     in databse    │
   └──────────────────┘
       │          │
    False        Ture
       │          │
       │          ▼
       │    ╱──────────╲
       │   ╱  Output    ╲
       │  ╱    main      ╲
       │ ╱     menu       ╲
       │ ╲──────────────╱
       │          │
       │          ▼
       └────────▶ ( End )
```
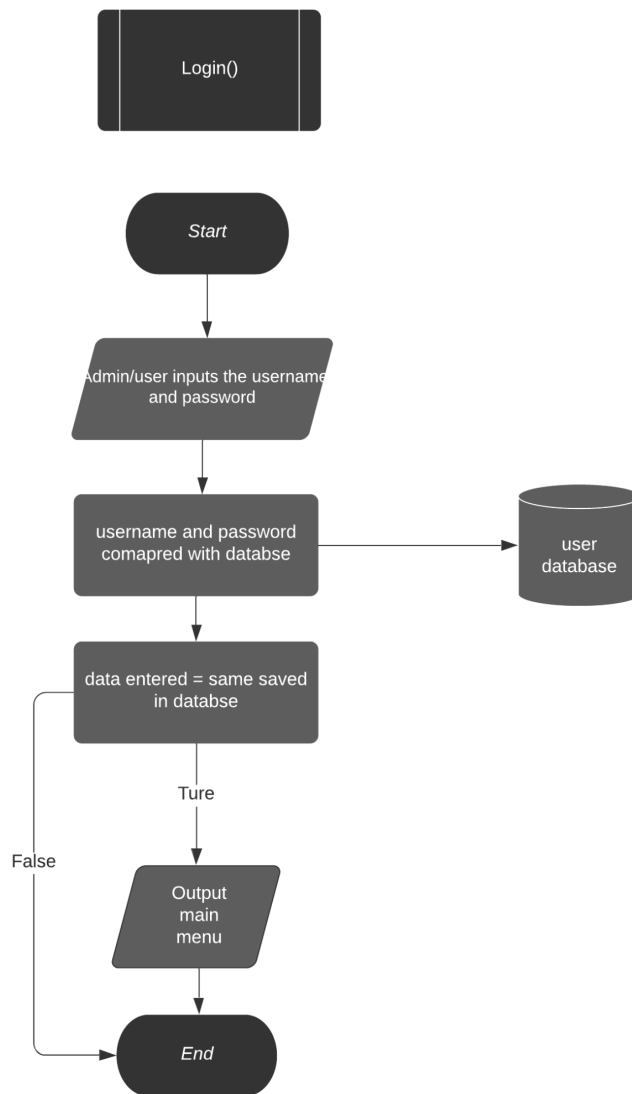
The login to a user account function outputs a login screen where the
user/admin is required to enter the username and password that was saved in
the database after they created their account. If the data entered matches
then they are presented with the main menu screen.

F1.3 Logout from a user account

```
        logout()


         Start
           |
           v
  Output welcome screen
           |
           v
          End
```
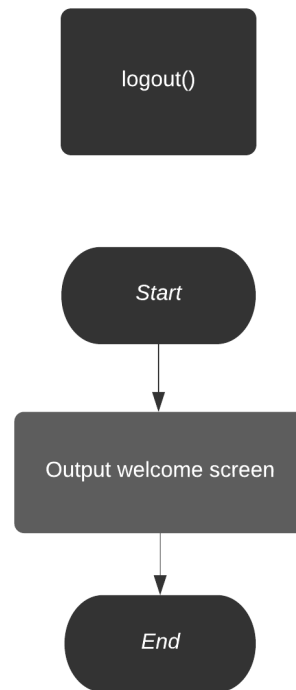
The logout function is just a follow up option when the user or admin logs in successfully. When the user/admin is logged in into his/her account and if wish to logout when they are finished with all their operations while using the system. As a result by choosing the logout option the user/admin can logout to the welcome screen.

F1.4 Delete user account
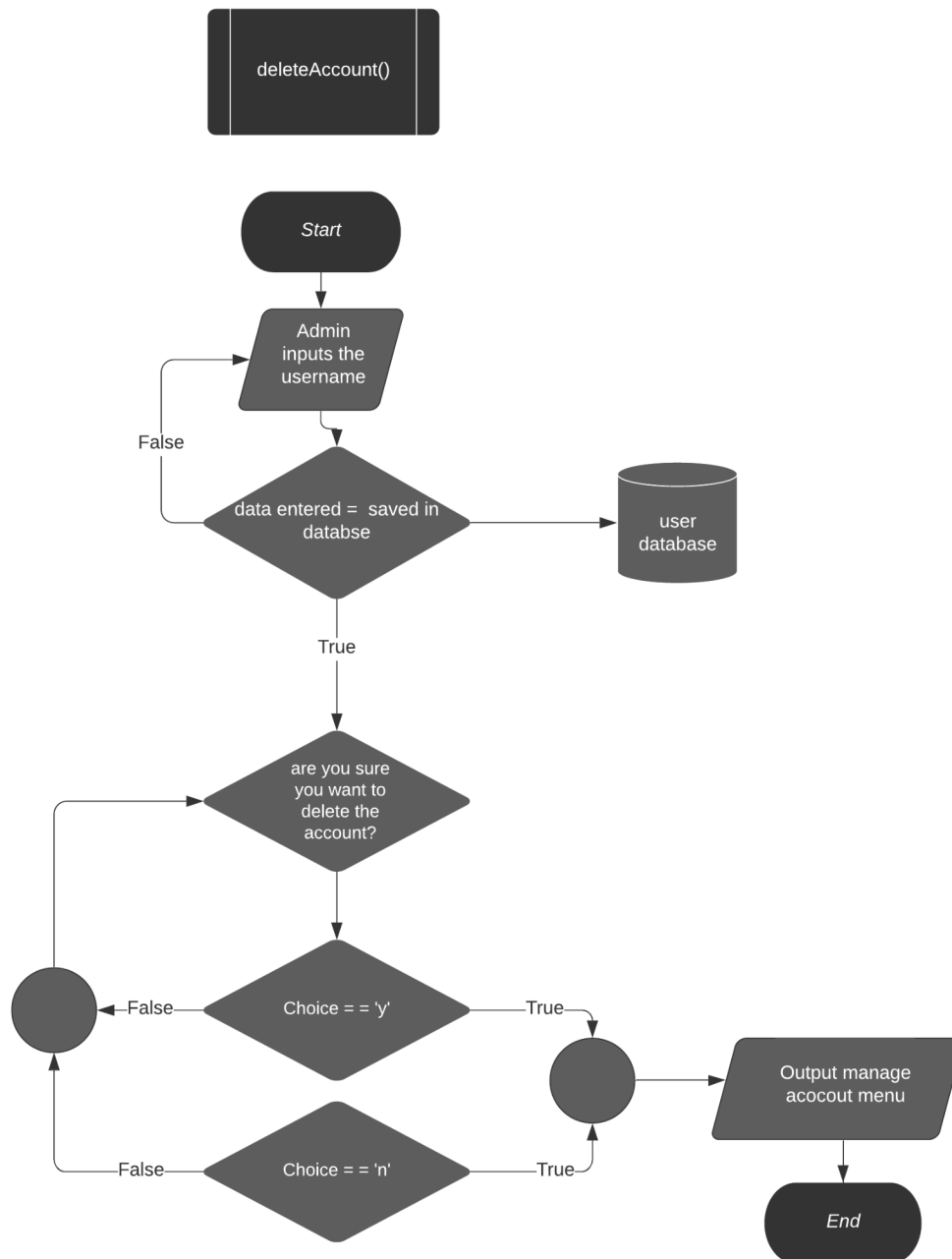
The delete user account function requires the Administrator to be logged in, as the normal user, in this case a buyer, cannot delete any of the accounts created and saved in the database. When the admin chooses the delete user account option the admin is required to enter the username if a match is found the program will delete the account when the admin confirms he wants to by choosing yes, then the account will be deleted from the database and users data will be displayed deleted as account status.

F1.5 Change user password
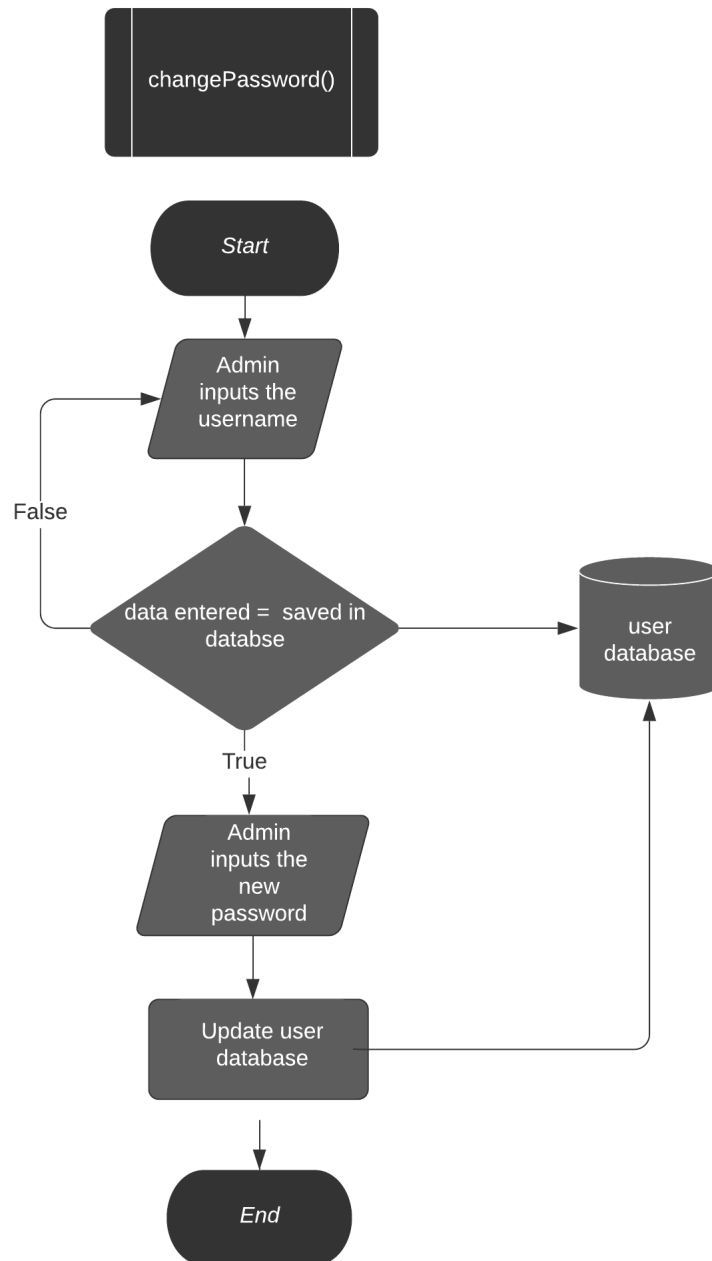
The change password function requires the Administrator to be logged as the normal user, in this case a buyer, cannot change the passwords for any accounts saved in the database. By choosing the change password option the admin is required to enter both username and new password if the password is not saved in the database before the process should be successful if the password does not meet the requirements will have to be more creative with it.

F2 Data file reader

F2.1 Load data file

The LoadDataFile() function get's the user's data filename from the database, and reads the data from the file. The reading sequence is as follows: number of columns, column names, computability of columns, number of rows, row datas.

F2.2 Save as
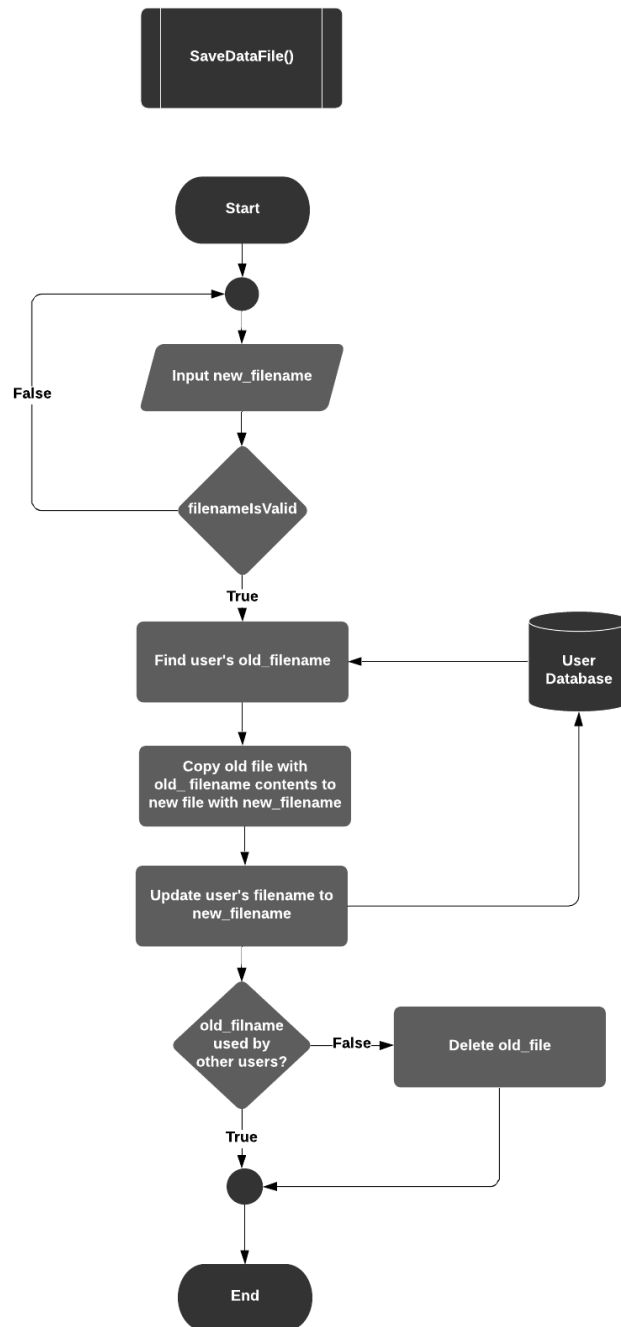
The SaveDataFile() function prompts the user to input the new filename and checks if the filename is valid (doesn't contain invalid characters and is not the same as system files). If so, the program proceeds to get the user's old filename, copies over the contents of the old filename to another file with the new filename, and then updates the user's filename to the new one. The old file with the old file name will be deleted if no other user is using it.

F2.3 Save report

```
ReportTXT()
```

```
Start
```

```
Output
Report
menu
```

Flase

```
Choice = = Generate
Text Report (.txt)
```

True

```
report file
created with
the data
```
→
```
data
```

```
Output
Report in
txt file
```

```
End
```
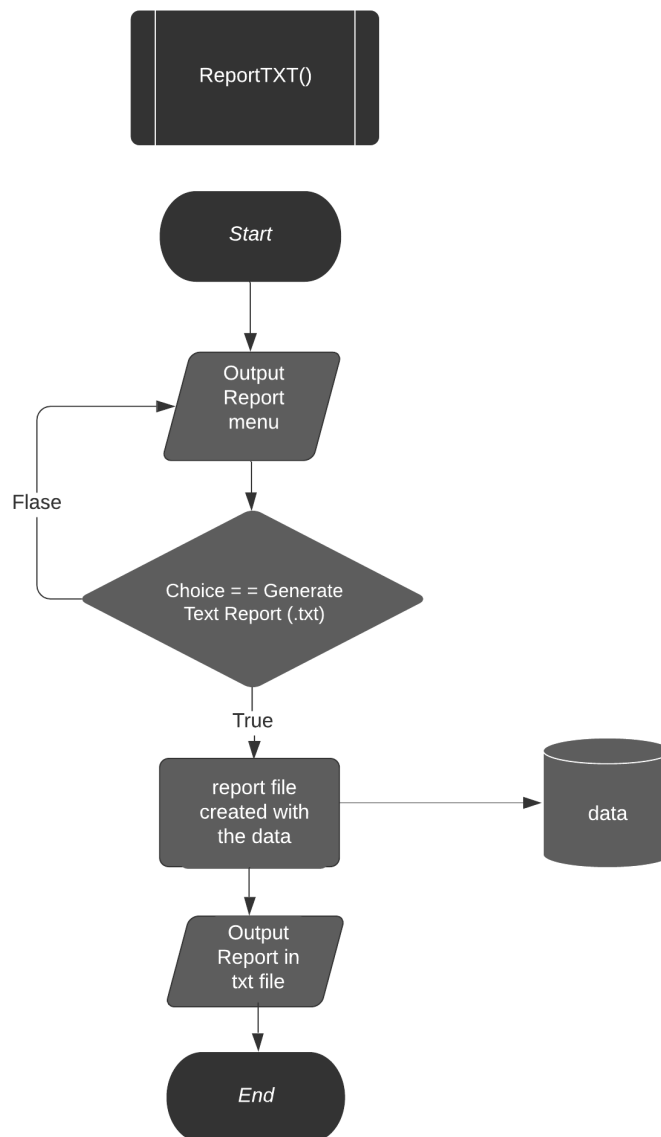
The Save report function has two parts txt and HTML in this case it is the txt format, which requires the user(buyer) to be logged in, this functions grants all the information calculated from the data file based on the every statiscal analysis function and puts the values found based on the user's choice while doing the statistical analysis calculations and stores them all in a txt file which can be view in an external window like notepad or a normal txt file.

F2.4 Save report HTML
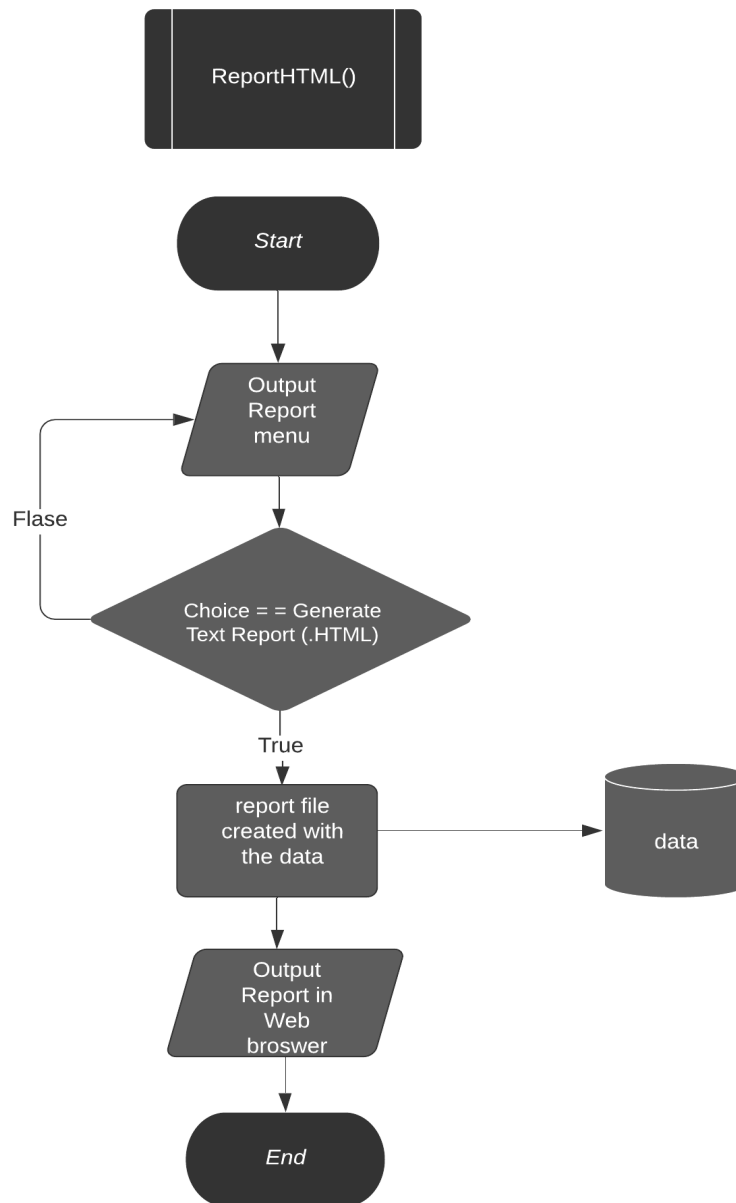
The Save report function has two parts txt and HTML in this case it is the HTML format, which requires the user(buyer) to be logged in, this functions grants all the information calculated from the data file based on the every statiscal analysis function and puts the values found based on the user's choice while doing the statistical analysis calculations and stores them all in a HTML file that can be viewed instantly after choosing the option in your default system browser which may defer for everyone.

F2.5 Log


```
                        logs()
```

Start → user logged in → user makes a descision

True → update log file

False → user logs out

Flase

update log file → (circle) → log database

user logs out → True → Exit and update log file → End

Exit and update log file → (circle)
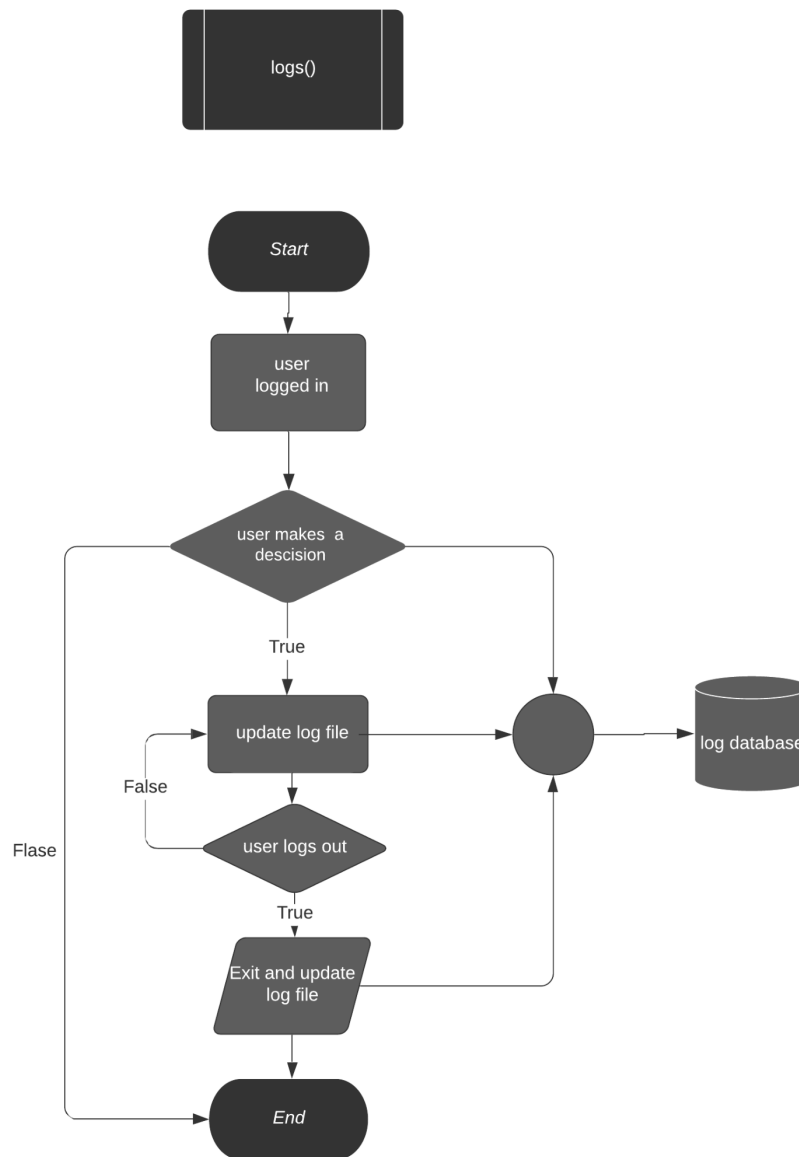
The logs function is quite simple as it saves all the actions or operations done by the user from when he logs in till he logs out the file updates automatically when the user logs out of the system, the format and what to expect can be viewed in the file or the presentation.

F3 Statistical Computation

F3.1 Find Minimum
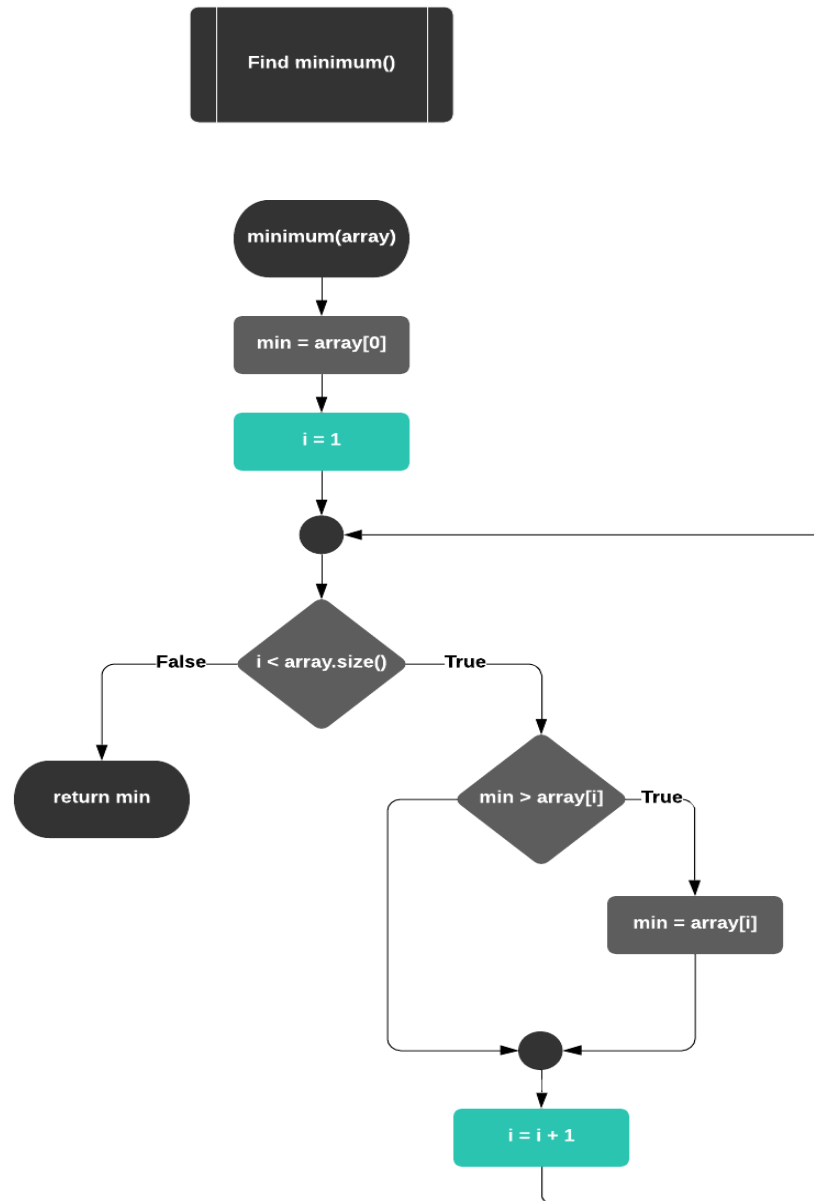
The minimum() function takes in an array and sets the first element of the array to be the minimum. The array is then looped through, starting from the second element to the end. Throughout that loop, if the prior set minimum value is larger than another element in the array, the maximum value is then changed to be the value of that smaller element.

F3.2 Find Maximum

```mermaid
Find maximum()
    ↓
maximum(array)
    ↓
max = array[0]
    ↓
i = 1
    ↓
i < array.size()  --False--> return max
    --True-->
max < array[i]  --True--> max = array[i]
    ↓
i = i + 1
```
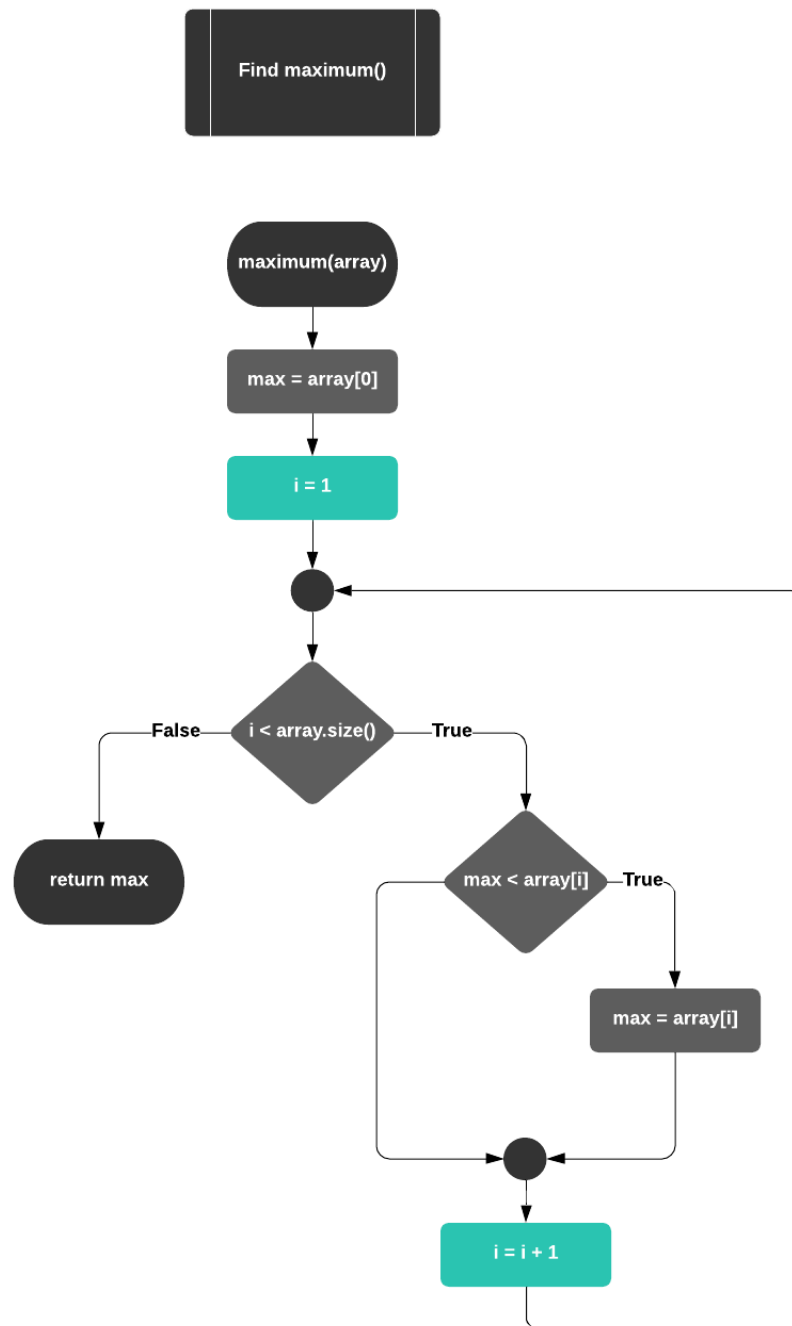
The maximum() function takes in an array and sets the first element of the array to be the maximum. The array is then looped through, starting from the second element to the end. Throughout that loop, if the prior set maximum value is smaller than another element in the array, the maximum value is then changed to be the value of that larger element.

F3.3 Median
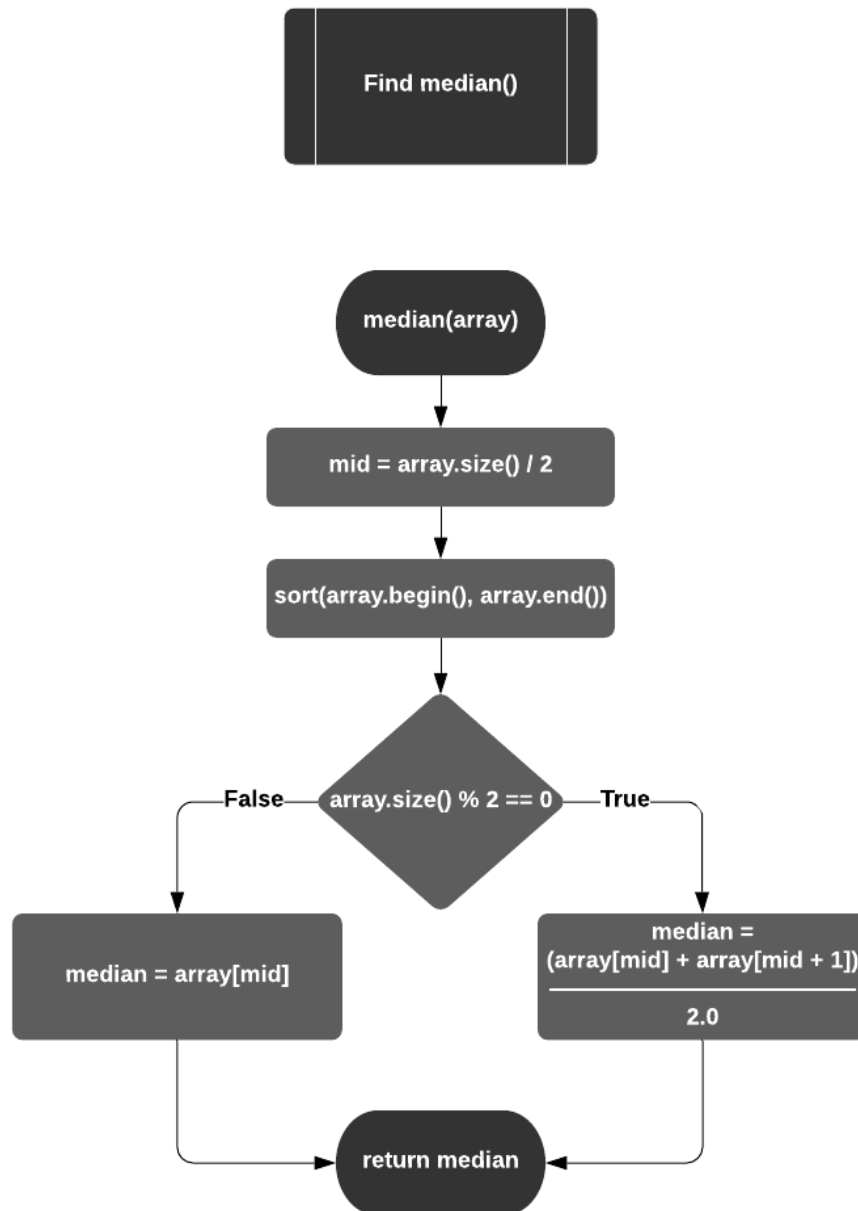
The median() function takes in an array and finds the index of the middle element by dividing the number of elements in the array by 2 and flooring it. The array is then sorted in ascending order. If the number of elements in the array is odd, then the value at the array of the middle index is the median. Else, if the number of elements in the array is even, then the median is the sum of the values at the array of the middle index and the value next to it, divided by 2.

F3.4 Mean
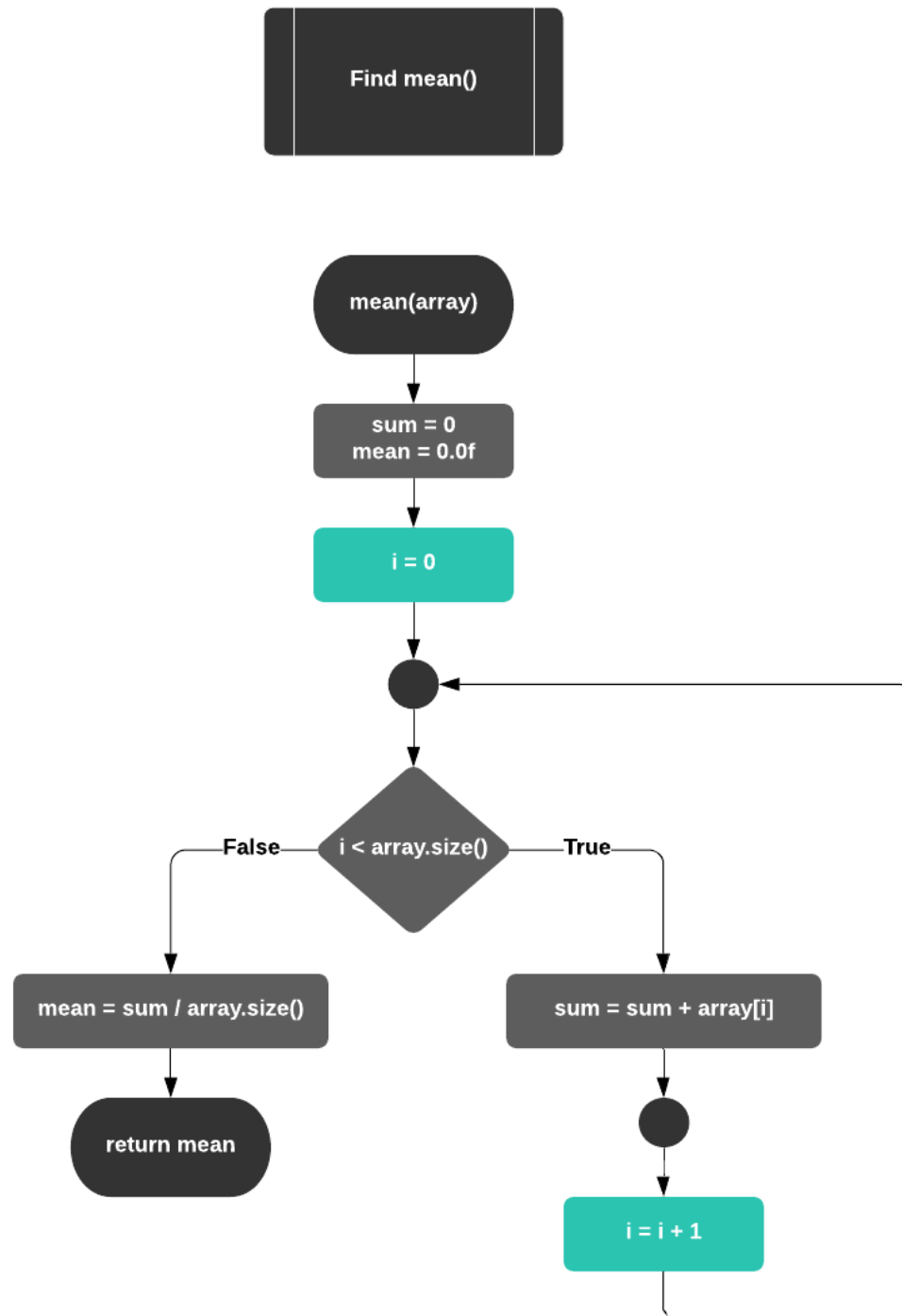
The mean() function takes in an array and loops through it to accumulate the sum of each element in the array. The mean is then calculated by dividing the sum of elements in the array by the number of elements in the array.

F3.5 Variance

```
                    ┌─────────────────────┐
                    │   Find variance()   │
                    └─────────────────────┘
                               │
                    ┌─────────────────────┐
                    │   variance(array)   │
                    └─────────────────────┘
                               │
                    ┌─────────────────────┐
                    │   n = array.size(), │
                    │     mean = 0.0f,    │
                    │   variance = 0.0f,  │
                    │     sdev = 0.0f     │
                    └─────────────────────┘
                               │
                    ┌─────────────────────┐
                    │     Find mean()     │
                    └─────────────────────┘
                               │
                    ┌─────────────────────┐
                    │        i = 0        │
                    └─────────────────────┘
                               │
                              ( )
                               │
        False ───────────    ◇ i < n ◇    ─────────── True
          │                                              │
┌─────────────────────────┐          ┌───────────────────────────────────┐
│ variance = sdev / (n-1) │          │ dev = (x[i] - mean) * (x[i] - mean)│
└─────────────────────────┘          └───────────────────────────────────┘
          │                                              │
┌─────────────────────────┐          ┌───────────────────────────────────┐
│      return variance     │          │          sdev = sdev + dev        │
└─────────────────────────┘          └───────────────────────────────────┘
                                                         │
                                      ┌───────────────────────────────────┐
                                      │             i = i + 1             │
                                      └───────────────────────────────────┘
```
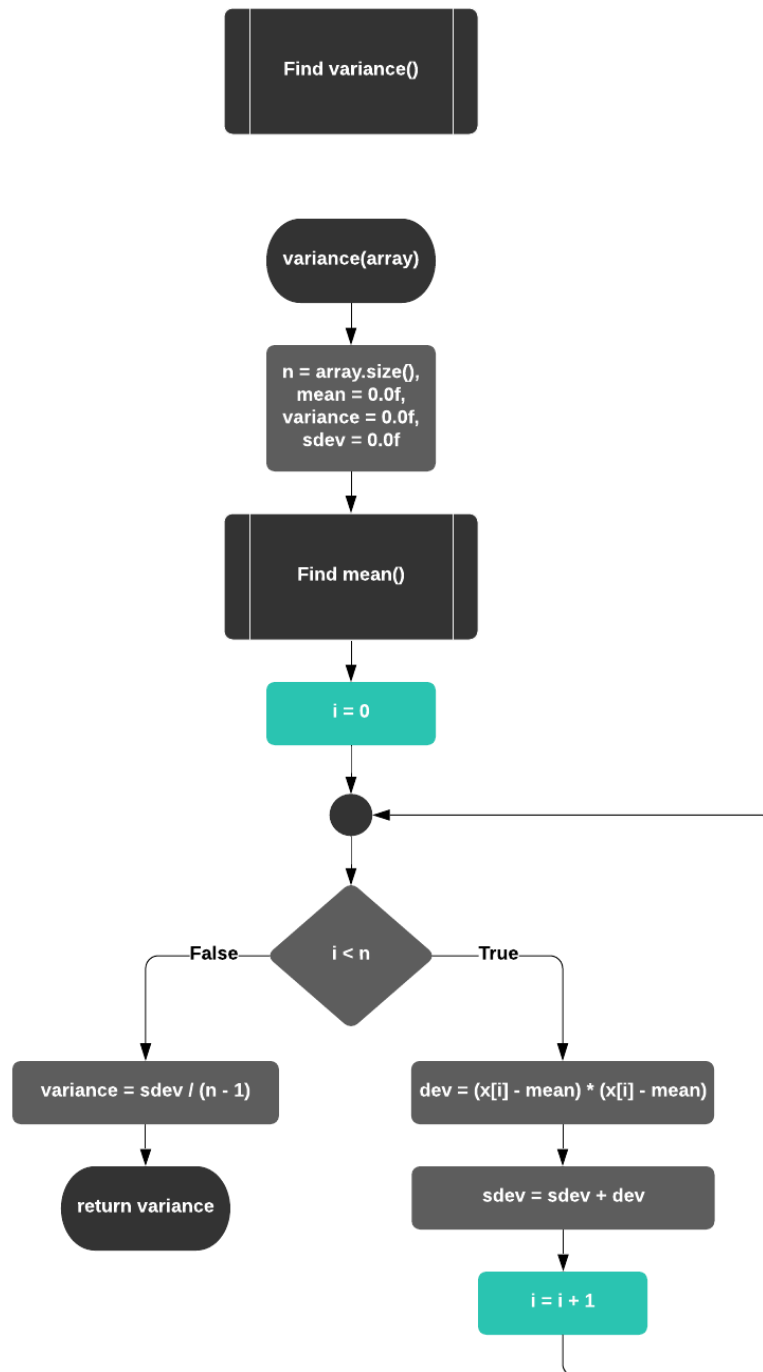
The variance() function takes in an array which is then used to find the mean
of the array. The function then loops through the array to calculate the sum
of each value subtracted by the mean value, to the power of 2. That value is
then divided by the number of elements minus 1 to get the variance.

F3.6 Standard Deviation

```
                    Find
              standard_deviation()
```

```
              standard_deviation(array)
                        |
                        v
                  variance = 0.0f,
               standard_deviation = 0.0f
                        |
                        v
                  Find variance()
                        |
                        v
           standard_deviation = sqrt(variance)
                        |
                        v
               return standard_deviation
```
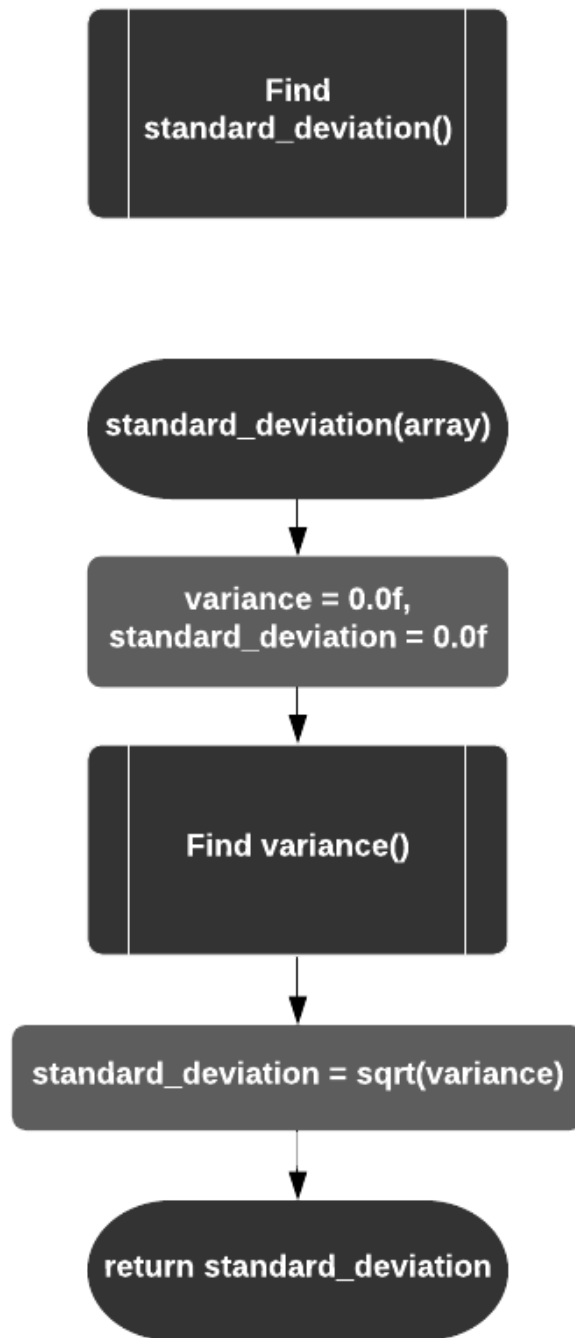
The standard_deviation() function takes in an array which is then used to find the variance of the array first. The standard deviation is then calculated by the square root of the variance.

F3.7 Correlation between any selected 2 columns

```
          Find
   pearsons_correlation()
```

```
pearsons_correlation(array1, array2)
```

```
          n = array.size(),
            meanX = 0.0f,
            meanY = 0.0f,
            devX = 0.0f,
            devY = 0.0f,
           sdevXY = 0.0f,
        sdevXsquare = 0.0f,
        sdevYsquare = 0.0f,
    pearsons_correlation = 0.0f,
```

```
       Find mean()
```

```
i = 0
```

i < n

False →

```
persons_correlation =
         sdevXY
    ──────────────────────
sqrt(sdevXsquare * sdevYsquare)
```

```
return pearsons_correlation
```

True →

```
devX = array1[i] - meanX
devY = array2[i] - meanY
```

```
sdevXY = sdevX + (devX * devY)
```

```
sdevXsquare = sdevXquare + (devX * devX)
sdevYsquare = sdevYquare + (devY * devY)
```

```
i = i + 1
```

$$r = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum (x_i - \bar{x})^2 \sum (y_i - \bar{y})^2}}$$

Pearson's Correlation Formula

The pearsons_correlation() function takes 2 arrays. The mean of the first and second array is found. Both arrays are then looped simultaneously to obtain the **devX** (value of the current element in the first array subtracted by the mean of the first array) and **devY** (value of the current element in the second array subtracted by the mean of the second array). The product of **devX** and **devY** are then added to **sdevXY** (sum of the product of **devX** and **devY**). The square of **devX** is added to **sdevXsquare** (sum of the square of **devX**) and the square of **devY** is added to **sdevYsquare** (sum of the square of **devY**). In the end, the **pearsons_correlation** is calculated by dividing **sdevXY** by the square root of (**sdevXsquare * sdevYsquare)**.

F3.8 Distinct data members

```mermaid
flowchart TD
    A[Find distinct_data()] --> B(distinct_data(array1, array2))
    B --> C[Declare distinct_array]
    C --> D[i = 0]
    D --> E((•))
    E --> F{i < array1.size()}
    F -- False --> G(return distinct_array)
    F -- True --> H[j = 0]
    H --> I((•))
    I --> J{j < array2.size()}
    J -- False --> K[i = i + 1]
    J -- True --> L{array1[i] == array2[j]}
    L -- False --> M[j = j + 1]
    L -- True --> N[distinct_array.push_back(array[i])]
```
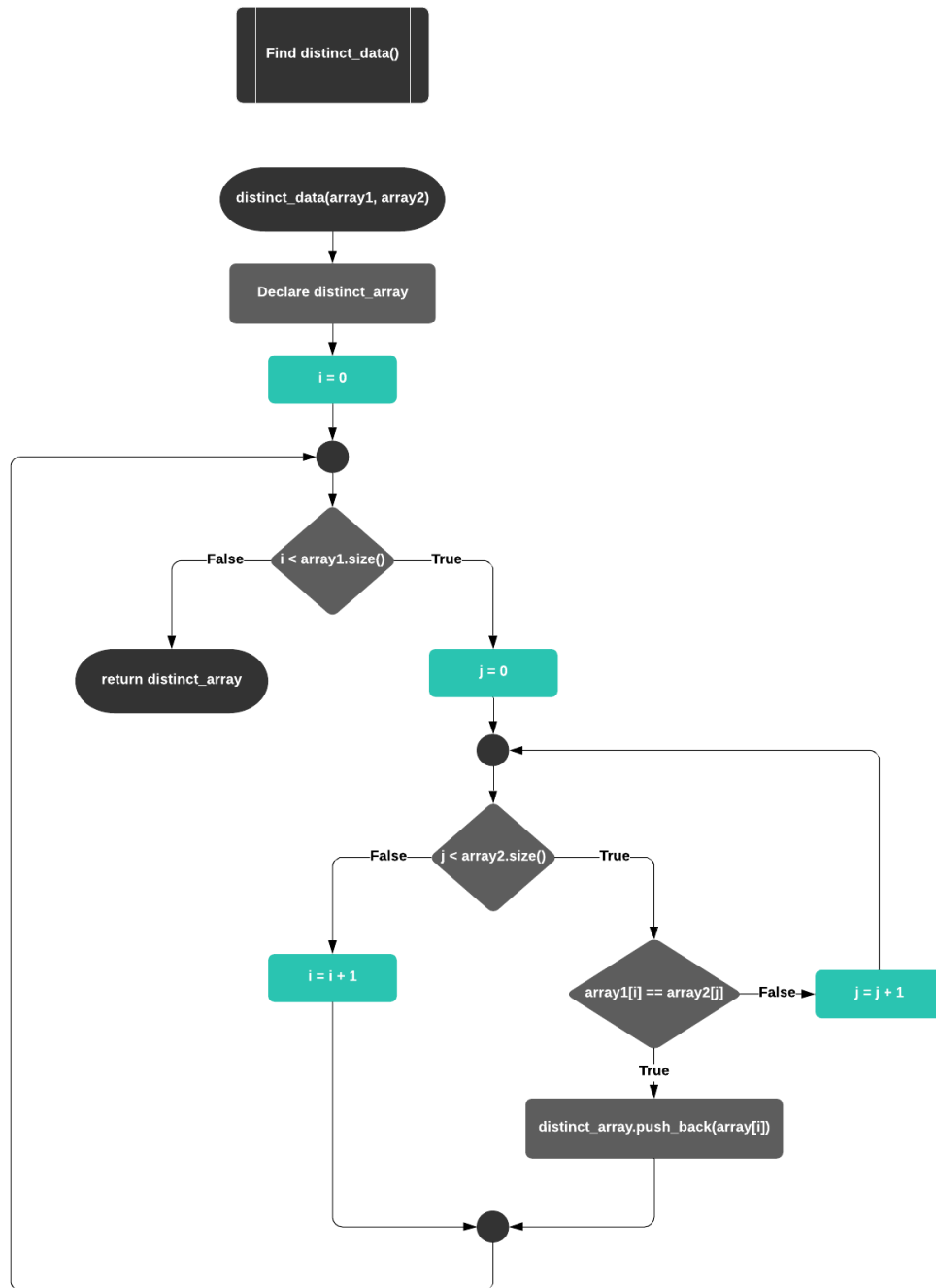
The distincy_data() function takes in 2 arrays. The function then loops through the first array and compares the element of the current iteration to the second array by looping through it too. If the elements are the same, then that element is added to a separate array which will be returned in the end.

F3.9 Plot A histogram (text mode)

```
┌─┤           Histogram()           ├─┐
```

```
          ╭──────────────────╮
          │  histogram(array)  │
          ╰──────────────────╯
                    │
                    ▼
          ┌──────────────────┐
          │  valueArray = array, │
          │   Declare valueCount │
          └──────────────────┘
                    │
                    ▼
     ┌──────────────────────────────┐
     │ sort(valueArray.begin(), valueArray.end()) │
     │ reverse(valueArray.begin(), valueArray.end()) │
     └──────────────────────────────┘
                    │
                    ▼
     ┌─┤    removeSortedDuplicates()    ├─┐
                    │
                    ▼
     ┌─┤        Find valueCount()        ├─┐
                    │
                    ▼
          ╱──────────────────╱
         ╱   Show Histogram   ╱
        ╱──────────────────╱
```

```
                  ┌─┐              ┌─┐
                  │ │ removeSortedDuplicates() │ │
                  └─┘              └─┘

              ( removeSortedDuplicates(array) )
                              │
                              ▼
                        ┌──────────┐
                        │   i = 0  │
                        └──────────┘
                              │
                              ▼
                             ● ◄──────────────────────────┐
                              │                            │
                              ▼                            │
        False──────────────◄ i < n ►──────────True        │
           │                                    │          │
           │                                    ▼          │
           │                       False──◄ array[i] == array[i-1] ►──True
           │                          │                         │      │
           │                          │                         ▼      │
           │                          │        ┌──────────────────────────┐
           │                          │        │ array.erase(values.begin()+i) │
           │                          │        └──────────────────────────┘
           │                          │                         │
           │                          │                         ▼
           │                          │                  ┌──────────┐
           │                          │                  │ i = i - 1 │
           │                          │                  └──────────┘
           │                          │                         │
           │                          └────────────► ● ◄────────┘
           │                                         │
           │                                         ▼
           │                                  ┌──────────┐
           │                                  │ i = i + 1 │───────┘
           │                                  └──────────┘
           │
           └────────────► ( End )
```

removeSortedDuplicates()

removeSortedDuplicates(array)

i = 0

i < n

False   True

array[i] == array[i-1]

False   True

array.erase(values.begin()+i)

i = i - 1

i = i + 1

End

```mermaid
Find valueCount()

valueCount()
  ↓
i = 0
  ↓
  ●
  ↓
i < valueArray.size()
  ├─ False → End
  └─ True → count = 0
              ↓
            j = 0
              ↓
              ●
              ↓
            j < array.size()
              ├─ False
              └─ True → valueArray[i] == array[j]
                          ├─ False → j = j + 1
                          └─ True → count = count + 1
                                      ↓
                                      ●
                                      ↓
                          valueCount.push_back(count)
                                      ↓
                                  i = i + 1
```

The histogram() function takes in an array, copies it, sorts it in descending order and removes the duplicates in it. The function then loops through the copied and modified array, and then loops through the original array and accumulates the number of times the current iteration element of the modified array is found in the original array. The frequency/count of each value is stored in another array and is parallel to the modified array. The modified array and the array which stores the frequency of each element are then used to construct the histogram.

F4 Menu System

F4.1 Main page

```
                              ┌──────────────┐
                              │  adminMenu() │
                              └──────────────┘

                    ┌─────────┐
                    │  Start  │
                    └─────────┘
                         │
                         ●
                         │
                    ╱─────────╲
                   │ Show options │
                    ╲─────────╱
                         │
                    ╱─────────╲
                   │ Input choice │
                    ╲─────────╱
                         │
                      ◇ choice == 'l' ◇ ──Yes──► logout() ─────┐
                         │                                      │
                       False                                    │
                         │                                      │
                      ◇ choice == '1' ◇ ──True──► RegisterAccount() ──●──► End
                         │                                      │
                       False                                    │
                         │                                      │
                      ◇ choice == '2' ◇ ──True──► ManageAccount() ────┘
                         │
                       False
                         │
                    ╱─────────╲
                   │ Show error │
                    ╲─────────╱
```
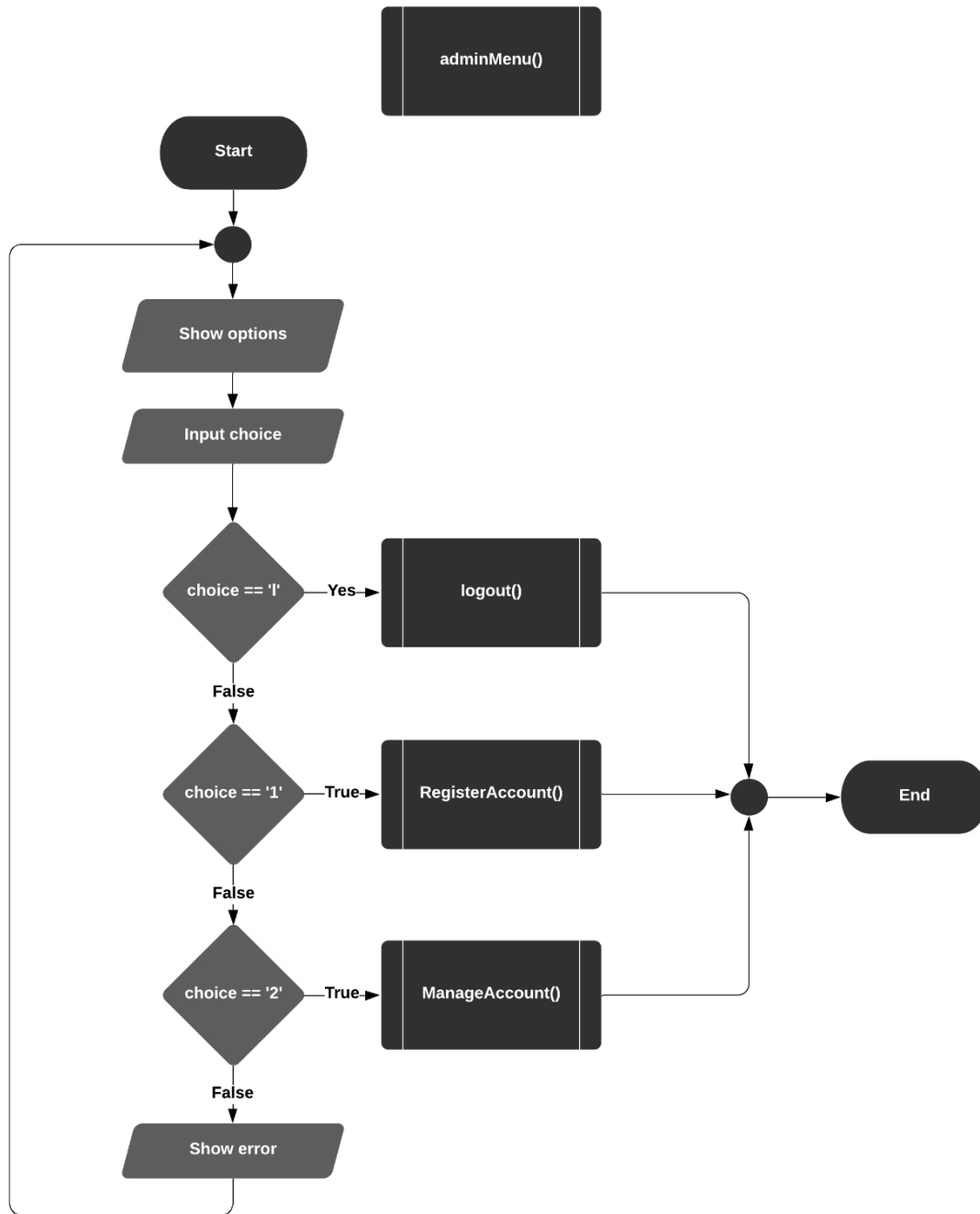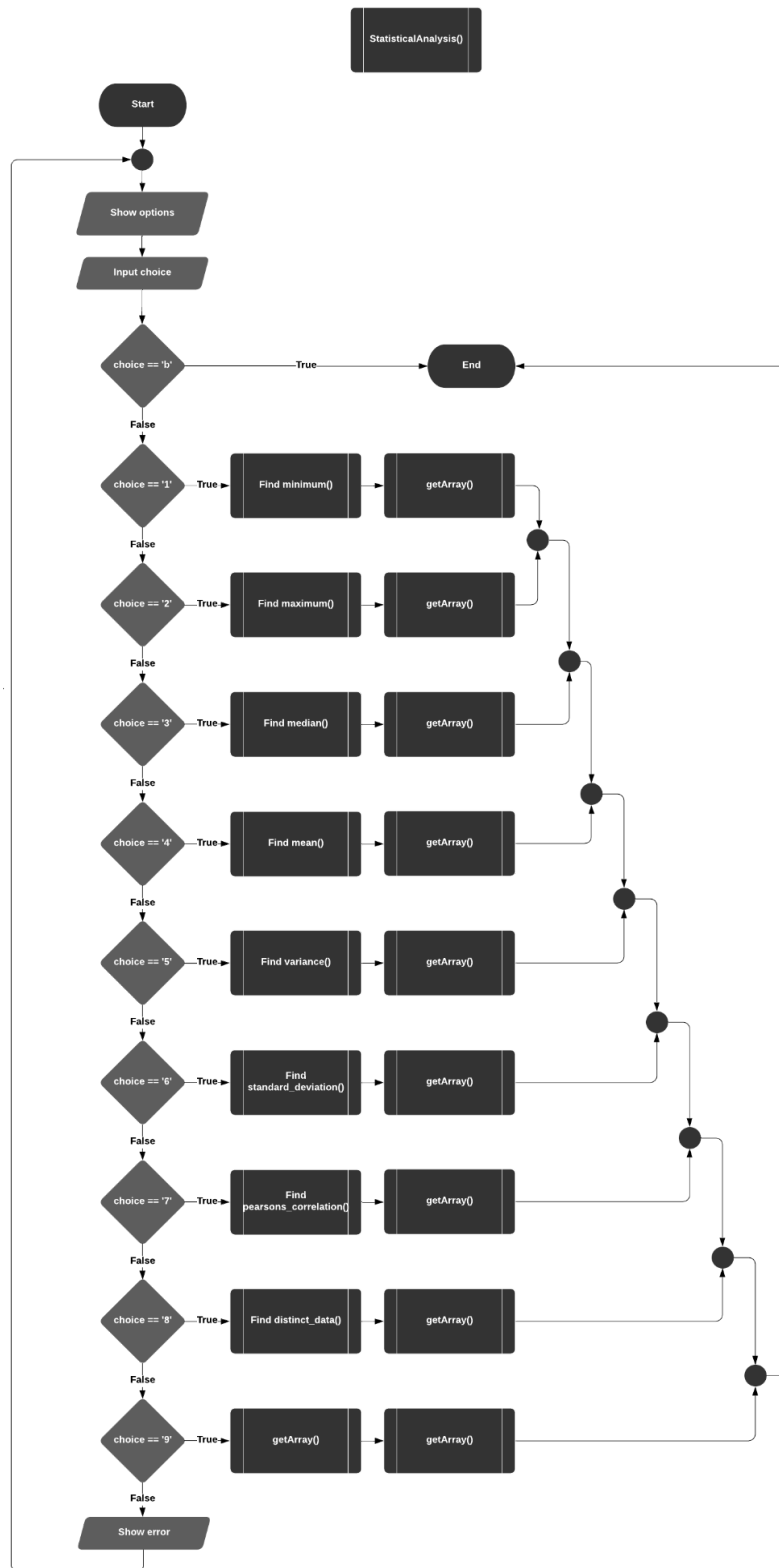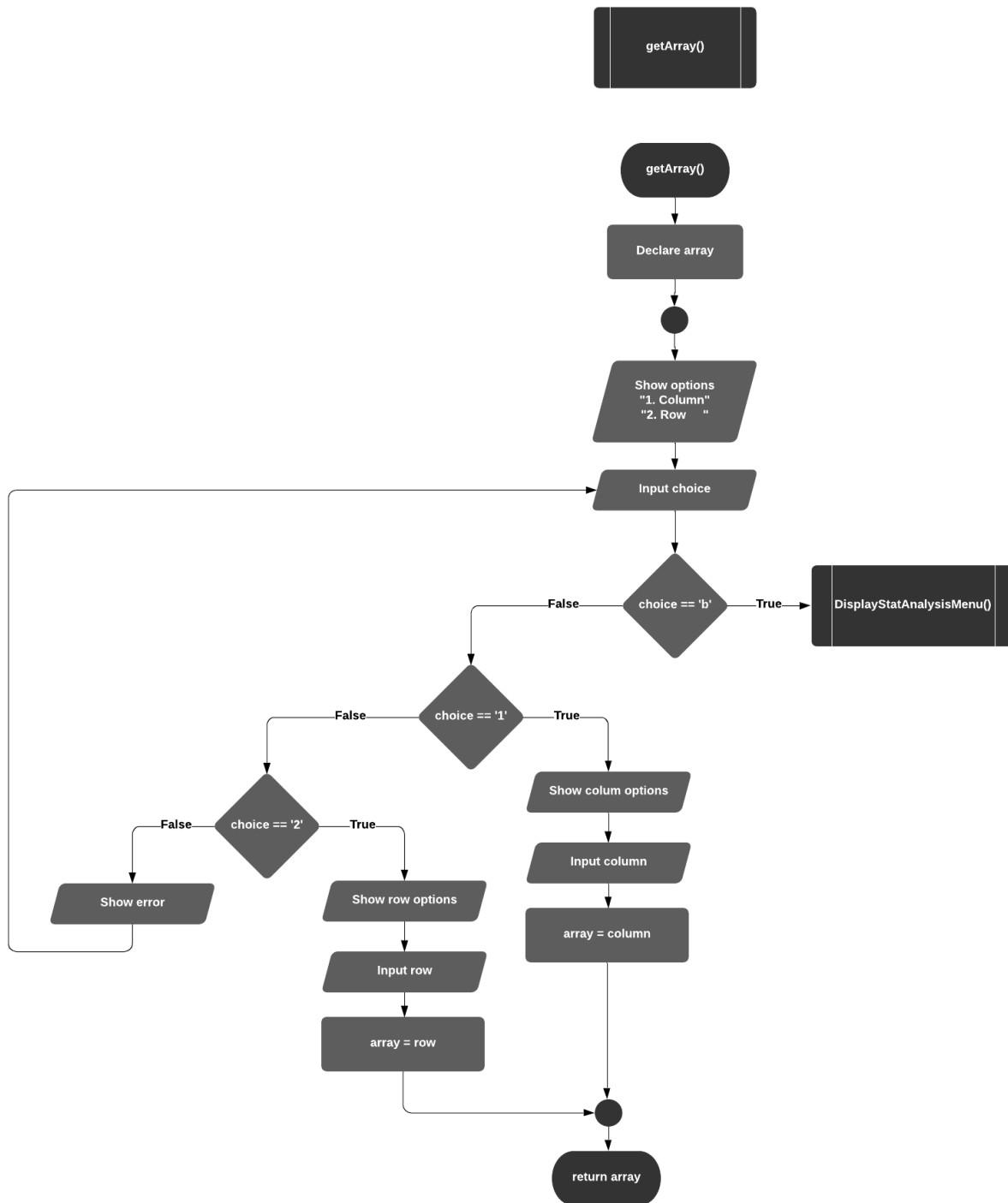
The adminMenu() displays all the admin-related functionalities for them to pick. If the choice is 1, 2, or 3, the respective functions will be called, else an error will be shown and the admin is prompted to enter their choice again.

```mermaid
flowchart
    userMenu()

    Start --> o((•))
    o --> ShowOptions[/Show options/]
    ShowOptions --> InputChoice[/Input choice/]
    InputChoice --> c1{choice == 'l'}
    c1 -->|Yes| logout[logout()]
    c1 -->|False| c2{choice == '1'}
    c2 -->|True| StatAnalysis[StatAnalysis()]
    c2 -->|False| c3{choice == '2'}
    c3 -->|True| SaveDataFile[SaveDataFile()]
    c3 -->|False| c4{choice == '3'}
    c4 -->|True| Report[Report()]
    c4 -->|False| c5{choice == '4'}
    c5 -->|Yes| ManageAccount[ManageAccount()]
    c5 -->|False| ShowError[/Show error/]
    SaveDataFile --> End
```

The userMenu() displays all the buyer-related functionalities for them to pick. If the choice is 1, 2, 3, or 4, the respective functions will be called, else an error will be shown and the user is prompted to enter their choice again.

StatisticalAnalysis()

Start

Show options

Input choice

choice == 'b' —True→ End

False

choice == '1' —True→ Find minimum() → getArray()

False

choice == '2' —True→ Find maximum() → getArray()

False

choice == '3' —True→ Find median() → getArray()

False

choice == '4' —True→ Find mean() → getArray()

False

choice == '5' —True→ Find variance() → getArray()

False

choice == '6' —True→ Find standard_deviation() → getArray()

False

choice == '7' —True→ Find pearsons_correlation() → getArray()

False

choice == '8' —True→ Find distinct_data() → getArray()

False

choice == '9' —True→ getArray() → getArray()

False

Show error

```
                              ┌─────────────────┐
                              │   getArray()    │
                              └─────────────────┘

                                ╭─────────────╮
                                │  getArray() │
                                ╰─────────────╯
                                      │
                              ┌───────────────┐
                              │ Declare array │
                              └───────────────┘
                                      │
                                     ●
                                      │
                              ╱─────────────────╲
                             │   Show options    │
                             │    "1. Column"    │
                             │    "2. Row     "  │
                              ╲─────────────────╱
                                      │
        ┌─────────────────────────╱──────────╲
        │                        │ Input choice │
        │                         ╲──────────╱
        │                              │
        │                         ◇─────────────◇                    ┌───────────────────────────┐
        │              False    ◇  choice == 'b' ◇    True ────────→ │ DisplayStatAnalysisMenu() │
        │            ┌──────────◇                 ◇                    └───────────────────────────┘
        │            │           ◇───────────────◇
        │            │                  
        │      ◇─────────────◇
        │  False│ choice == '1'│  True
        │  ┌───◇              ◇───────┐
        │  │    ◇────────────◇        │
        │  │                     ╱───────────────╲
   ◇─────────────◇               │ Show colum options│
False│ choice == '2'│ True        ╲───────────────╱
┌───◇              ◇───┐                │
│    ◇────────────◇    │         ┌───────────────┐
│                      │         │ Input column  │
╱─────────────╲  ╱───────────────╲   └───────────────┘
│  Show error  │  │ Show row options│       │
╲─────────────╱  ╲───────────────╱  ┌───────────────┐
                       │            │ array = column│
                ╱───────────────╲   └───────────────┘
                │   Input row     │          │
                ╲───────────────╱           │
                       │                    │
                ┌───────────────┐           │
                │  array = row  │           │
                └───────────────┘           │
                       │                    │
                       └────────→ ● ←───────┘
                                  │
                           ╭─────────────╮
                           │ return array│
                           ╰─────────────╯
```

The Statistical Computation Menu displays all the statistical functionalities for users to pick based on the choice number. If the choice is valid, it performs the getArray() to get either the row data or column data or both based on the specific statistical computation chosen. Else it will show an error and start all over.
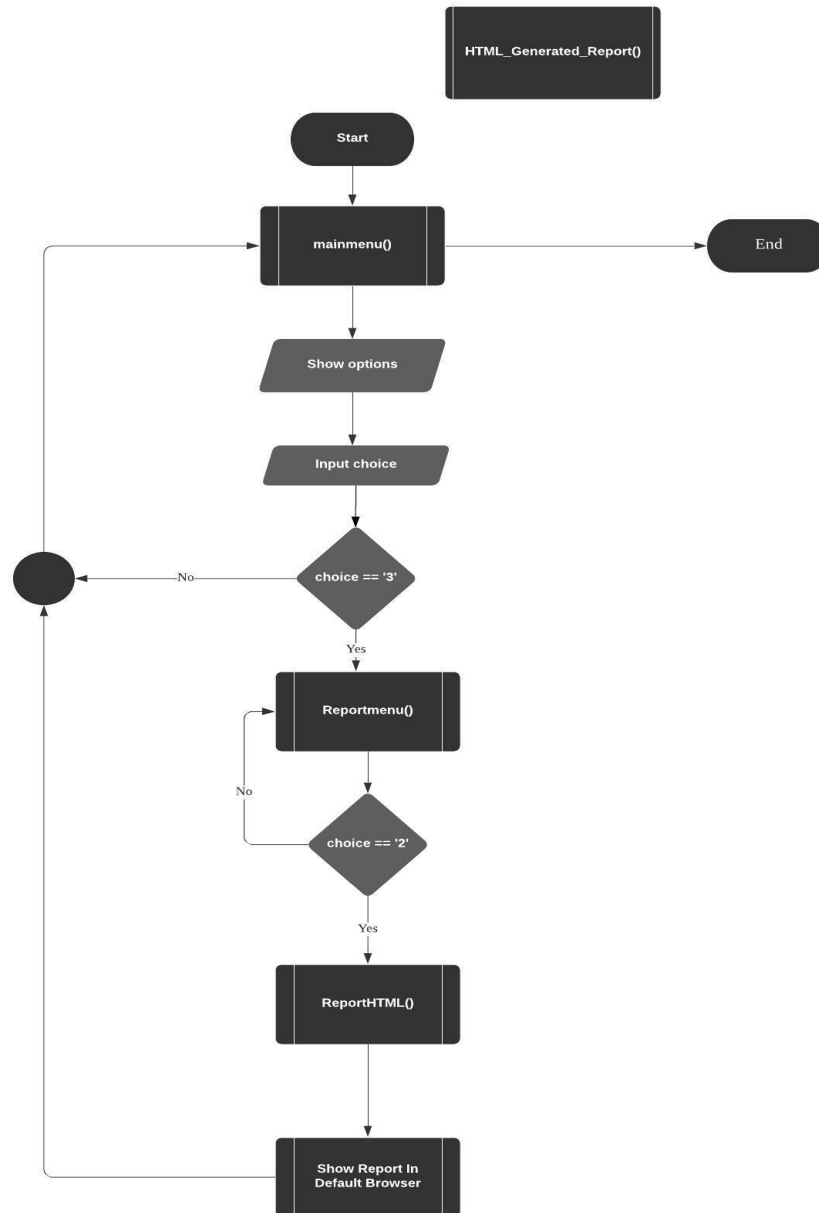
F4.3 Reports menu
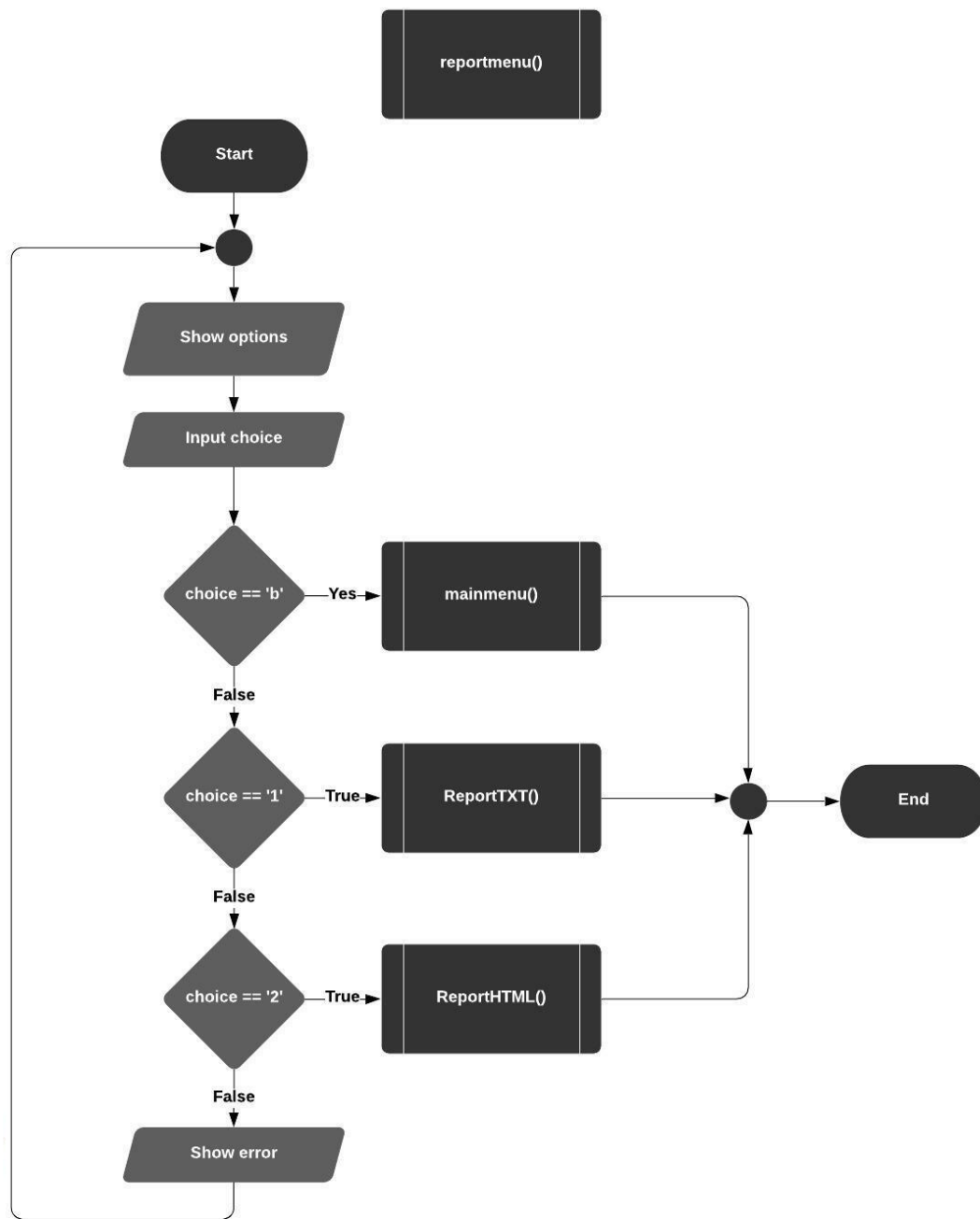
The Report Menu displays the option to go back, 'b', option to generate text report, '1', and option to generate HTML report, '2'. If none of these choices are input by the user, an error message will appear and the user is prompted to enter their choice once again.

F4.4 HTML generated report

The HTML generated report function starts with showing the main menu and asks the user to input the choice to generate a HTML or a text report. Once the user input option '2' the report will be generated in HTML format and automatically opened in your default browser to show all the statistical computation for each row and column.

F4.5 Error checking
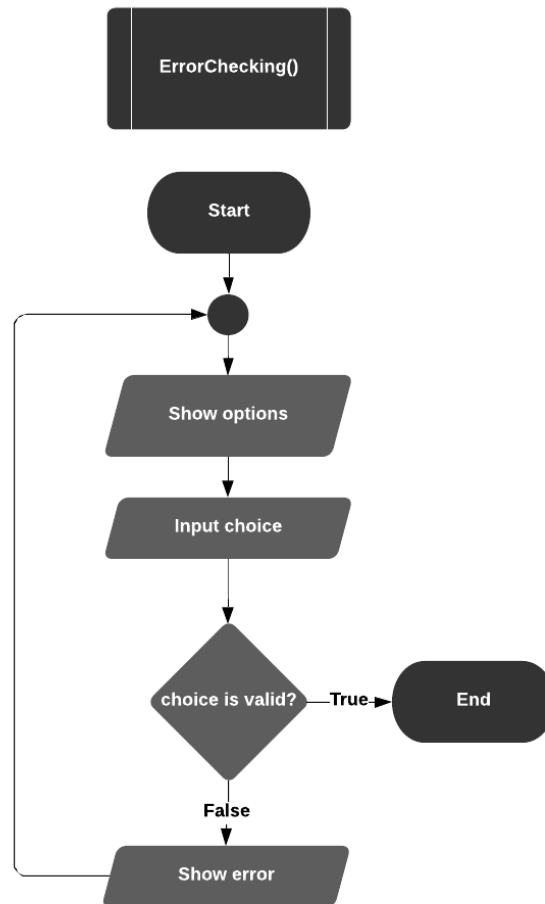
The ErrorChecking function basically incorporates into the do-while loop for each of the menus. If the user enters a selection that is not valid (not shown in the menu), the function will show an error message and loop back to asking an input based on the given choices shown in the menu.The loop will continue until the user enters a valid option when prompted to do so.

SAMPLE INPUT & OUTPUT GENERATED

Sample Input File:

Sample Output Generated:

```
   ____  _____  ____  ___  _____
  /_  \/  _/ __/_  _/ /_  \/  |/_  _/ /   |
 /// // /\_\// //   /// /// /\ \/// /\|
/ /_/ // / __/ /// /  /_/ / __ \/ / / __ |
/____/___//___//_(_) /____/_/  |_/_/ /_/  |_|


+-------------+ DIST. DATA MEMBER +-------------+

Data: Y Column
| 94 | 59 | 94 | 94 | 80 | 79 | 59 | 77 | 66 | 94 |

Distinct Data:
| 59 | 66 | 77 | 79 | 80 | 94 |


+---------------+-----------+
| Distinct Data | Frequency |
+---------------+-----------+
|            59 |         2 |
|            66 |         1 |
|            77 |         1 |
|            79 |         1 |
|            80 |         1 |
|            94 |         4 |
+---------------+-----------+


+------------------------------------------------+

Go Back <ENTER>
```

```
  __  _____ _____  ___   __  __
 ///// _/ __/_ _/ _ \/___/_ \  | / / \/ /
 / /_/ // / /\_\// / / / / /_//_/ /\ \ / //\_/ /
/ _  // / __/ / / /_/ / /_/ /  _, _/ __ \// / //
/_/ /_/___//___//_/  \___^___/_/ \_/_/ \_/_/  /_/


+------------------+ HISTOGRAM +------------------+

Data: X Column
| 17 | 12 | 13 | 16 | 15 | 16 | 14 | 18 | 16 | 19 |

Count        X
    1       12 |=
    1       13 |=
    1       14 |=
    1       15 |=
    3       16 |===
    1       17 |=
    1       18 |=
    1       19 |=


+------------------------------------------------+

Go Back <ENTER>
```

CONCLUSION

This project's purpose is to create a console-based, easy-to-use basic data analysis system. For this project, we've used C++ and a text file database.

We started off by gathering some ideas from other console-based programs to analyse its qualities to implement in our system and its weak points to avoid in our system. Some ideas were also gathered from the lectures, tutorials, and consultation hours, which aided us to build and improve the system.

From there, after a comprehensive analysis of the system's functional requirements, we moved to designing the system and its database. Once we had a rough idea of what we were building, it was time to delegate the work and start coding. These pre-works and phases have really eased the development process for this project.

The system's creation and complete implementation took place in the project's latter stages. We had tested each functionality and work by our team members to see if it was coherent with the design we had in mind. In the final stages of our development, we began combining our work, which didn't go as smoothly as we thought but we managed to pull through a fully functioning data analysis system.

One major obstacle in this project was with file handling and manipulations as that was the sole foundation of this data analysis program. Nevertheless, we still came through after learning about it in our lectures and also with the help of the Internet. In the end, we've gained a lot of knowledge and experience from problems and issues throughout this project which sets up a better understanding for the upcoming development projects.

Overall, the project has accomplished the aim we envisioned about a console-based, easy-to-use basic data analysis program. We believe we have done our best in many aspects of our approach to this project, from design to code clarity, resulting in a pleasing outcome for our system.

REFERENCES

1. C++ Library & Functions

   https://www.cplusplus.com/

2. C++ Date and Time Utilities

   https://en.cppreference.com/w/cpp/chrono

3. Eugene O'Loughlin, *"How To... Calculate Pearson's Correlation Coefficient (r) by Hand"*.

   https://www.youtube.com/watch?v=2SCg8Kuh0tE