Software Development Plan (PDF)

Application overview:

Purpose:

- My application will have two purposes'.
- Purpose one is to attempt to help the user improve their current mood if they are feeling down via a guide of science backed recommended activities.
- Purpose two is to record their moods with a daily journal / chart so they may be able to provide better quality feedback to their GP, thus eliminating the need to rely on inconsistent active recall and as a result receive a more accurate diagnosis, treatment plan.

Aim:

- It is aimed at those who suffer from a mental illness and the general public alike.

Reason:

- The reason i think this application would be beneficial to the user is because sometimes it can be difficult to think of enjoyable things when you're feeling low.
- Therefore this application will recommend healthy activities for you when you do not have a healthy perspective to do so yourself.
- Features such as a Mood Journal & Mood Chart will be included so the user can help their GP give a more accurate diagnosis.
- Charting their mood will allow the user and their GP to see patterns in their life and offer a way to monitor their own progress and possibly even identify triggers that decreased their level of mood throughout the week.
- Their are Australian mental health organisations that offer mood charts, journals and crisis kits like this but they are print out PDF's and no one want's to carry a chunky folded up paper chart in their wallet. The user needs something versatile like an "E-kit" and it needs to be personalized to them (like how a bot acts).

Usability - How will the user interact with my program?:

- Since the user may be in a negative state i want the UX to be as simple as possible with little to no thought needed. However because the assessment requires text input i will require the user to enter their name via text input. The rest of the features and menu will be navigated using keyboard directional keys.

Functional / Algorithmic Overview + Implementation Plan:

Name of the app'?: **"MH-Kit"**

Welcome message: *(Describe to the user what to expect so they can be familiarized with the features before the menu is displayed)*

1. Use (system clear) function so that a second run of the program will get a clean / fresh terminal

1. Use (system clear) function so that a second run of the program will get a clean / fresh terminal window
2. "Welcome   to   MH-Kit!" (make this a fancy ASCII title)

- Which ruby GEM for the title?: --->Figlet

1. (standard text) "Hi there! Welcome to MH-Kit."
2. **[space]** Use (puts)
3. "I am a virtual mental health toolkit that will help you get back on your feet when times get tough. Self help can be hard when your down in the dumps. I am here to try boost your mood with a guide of healthy activities when your not in the right frame of mind to think of such outlets yourself."
4. **[space]** Use (puts)
5. "MH-Kit also come's equipped with a (*feature one name*) & (*feature two name*) so you can help your GP give you a more accurate diagnosis. Charting your mood allows you to see patterns in your life."

Start of program:

1. (Need to prompt the user for their name for added personalisation)EG: "What is your name?"

- Use a combination of a (string) for prompt, (variable) called name to store the name and (gets.chomp) for input

1. Program asks "Hi [name] what would you like to do?"
2. *Displays menu*(with short **non-distracting** description in brackets of how to navigate / interact with the program).
3. **Priority of this feature: Non-Crucial**

Menu Options:

- Launch Activity Guide
- Mood Journal
- Weekly Mood Chart
- Mental Health FAQ's
- Exit
- **Priority of this feature: EXTREMELY Crucial - Make this the first goal!!!**
- **Time Deadline:** Start/ Tuesday night (8:00pm). Finish/ Wednesday 4th 4:00am

--------------------THREE FEATURES AND DESCRIPTIONS OF EACH------------------- + IMPLEMENTATION PLAN CONTINUED

How will i display the menu?

- **1st decision:** List of strings that include a number inside the string. Attached to an if else statement that says "(If 1 then do this option. Else If 2 then do this, etc., etc.)
- **2nd decision:** Create a class called menu.

- Use an array for menu options so that i can use index + 1 to number them.
- Prompt user to select a choice. Receive user input via gets.to_i.
- **Final decision:** TTY:DSL:MENU Class + use a hash for the menu items so that i can use the menu options (keys) to create values for those keys which will make creating the case statement below it easier. (Includes instructions for usability for menu for me as well! plus highlights the option the user is currently on thus aiding navigation even more, awesome!)
- How will the user return to the menu?: They wont need to. Write it so the menu keeps displaying/outputting underneath the application feature they are currently using. To do this use *loop do* but make sure to put != [exit value] so that if the user selects the exit option/value the menu is not displayed and therefore the program meets its end.

<u>How will the menu option contain functionality / actually do anything when the user selects an option?</u>

- Functionality sounds like a (method)
- First i need the menu options to be assigned to a method with a similar to same name of the 'feature name' for DEV readability purposes.
- Lots of options will make a messy if/else statement. Use a case statement, their good for checking against multiple conditions/options.
- Create the methods AFTER i have created the skeleton of the menu.
- Then create the features functionality via methods
- Then case statement to include the methods.

Feature 1 Description- Activity Guide:

- ***Priority of this feature: EXTREMELY-Crucial!!*** *1st course of action after menu is created! This is the main initial idea and focus of my application. It is the feature that is mostly responsible for achieving my applications purpose of improving the users mood.*
- **Time Based Deadline:** Finish By Wenesday 4th 11am

1. Upon selection of the activity guide menu option. The case statement should be called.
2. The case statement will match the menu-elements(activity guide) value (i define in the menu hash) to the value i will set in the case statement conditions.
3. Use puts[method_name] to then output the menu option once matched.
4. Activity guide will be displayed in a table using terminal-table(the one Matt showed us in class).
5. 1st column of cells to be the name of the activity.
6. 2nd column of cells to be the description / justification of why the activity was recommended to them(the user).

Feature 2 Description- Mood Journal:

- *> Priority of this feature:* ***Medium-Crucial***
- **Time Based Deadline:** Finish By Thursday 5th 3:00am

- Use the TTY-Slider for the user to input their daily mood rating of 1-10. To do this create a new instance of the TTY::Prompt class and assign it to a variable with a name relating to its purpose (for dev readability).
- Provide description of how the mood journal should be used above the slider in strings.
- Use the TTY-Prompts abilities to output easy to understand instructions along with it.
- Tell the user that the purpose of the mood journal is so that the data can be assigned to the weekly mood chart even though i haven't added this functionality yet due to **time restrictions.**
- If user inputs a mood rating of less then 5 then attempt to cheer them up with a list of inspirational quotes using strings.
- Elsif mood_rating > 5 then simply say something like "Well done! Your making great progress!" and provide feedback to the user that their mood rating has been recorded.

Feature 3 Description- Mood Chart:

- **> Priority of this feature: Medium-Crucial**
- **Time Based Deadline:** Finish By Thursday 5th 11:00am
- Decide if you will display mood stats in list format or chart format (chart format is easier to read)
- Display this mood chart in an easy to understand format.
- Use ASCII-chart gem for this.
- ASCII-chart gem only comes in Cartesian format so remove bottom values and assign strings to them instead which will be the days of the week.
- Removing bottom integer values will turn the chart format into the desired bar graph format. Easier to understand for the user.

Feature 4 Description- Mental Health FAQ Table:

- **> Priority of this feature: Least-Crucial.** This feature is an add-on if i have enough time after coding the others. It doesn't directly contribute to solving the problems my application is trying to solve but yet it is still a factor in improving the users mood / outlook and therefore not an unnecessary feature to be adding in.
- **Time Based Deadline:** Finish By Thursday 5th 9:00pm
- Should be the same process as the activity guide table.
- 1st column of cells to include Fact number.
- Second column of cells to contain the actual facts.
- Display referral to lifeline at the bottom of the table in error-red color.

Possible errors that could occur?

- code a second if/else statement into the case statement to display a string saying "You have already selected activity guide. Please choose another option." in red, if a user selects the same option twice.
- Update: Purged this condition due to the possibility of the user maybe wanting to select the same feature twice such as the mood journal in case they wanted to update their input or look at their chart again.

- Update: Eliminated all **ERRORS**.

User Interaction and Experience

How will the user interact with / use each feature?

- Construct each feature to be as simplistic and self explanatory as possible. Good UX needs no explanation or instructions.
- Include brief descriptions of the use of the feature inside of the correlating feature via string text above the feature itself. This should justify the reason the feature has been included. The actual inputs required to use it should be explained in non-distracting brackets beside the user-prompt.

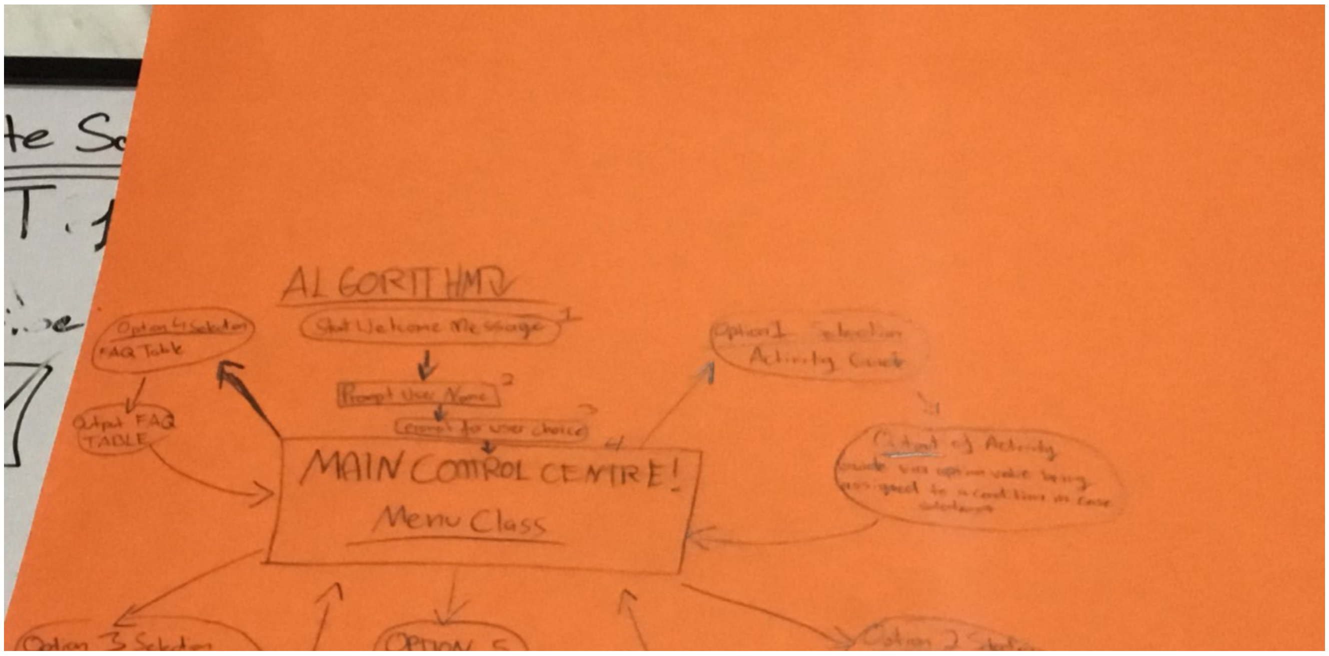
How will errors be handled by the application and displayed to the user?

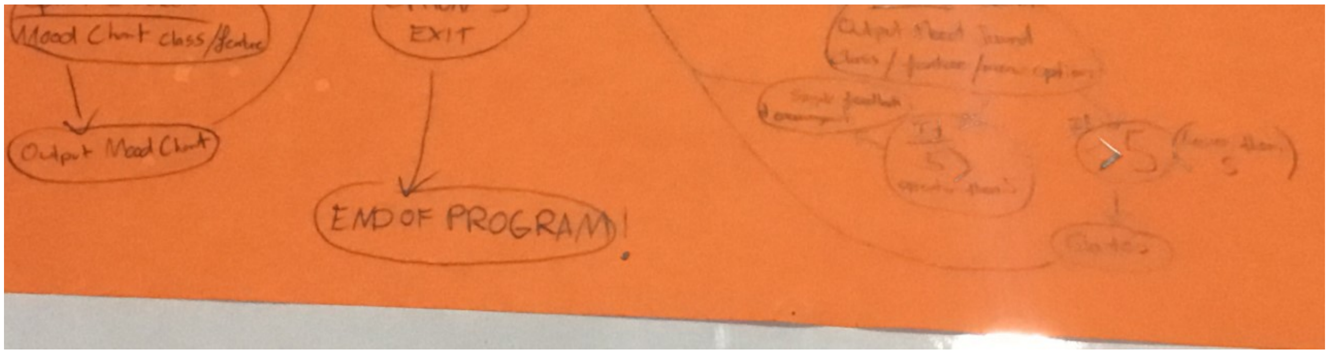
- Create the application so that there is no reason to display error feedback to the user.
- Make it impossible for an error to occur via re displaying the menu underneath each feature output.

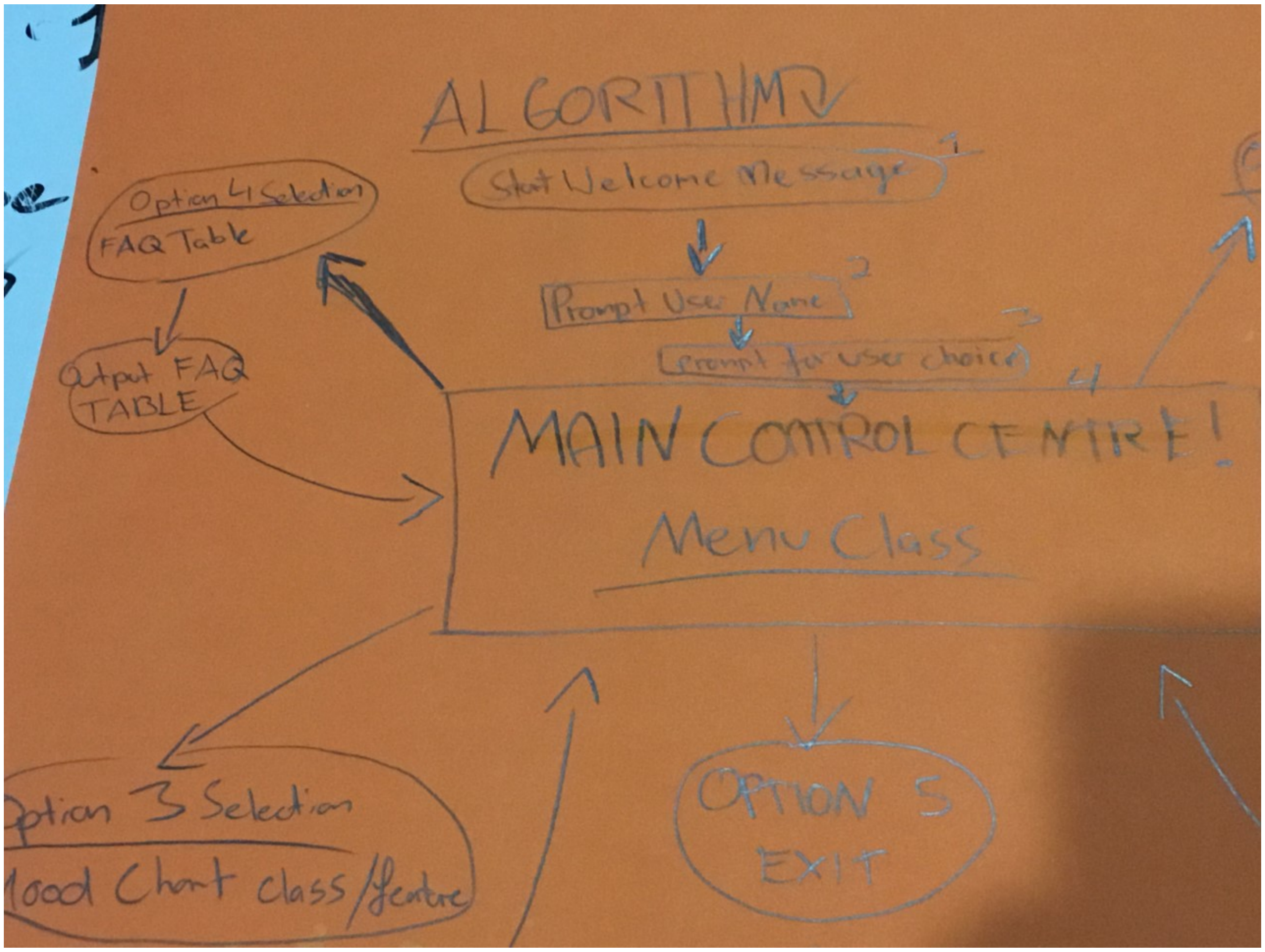
- Keep the conditions and code structure as simplistic as possible while still meeting the requirements of the assignment and exercising what i have learnt in class in the past two week.

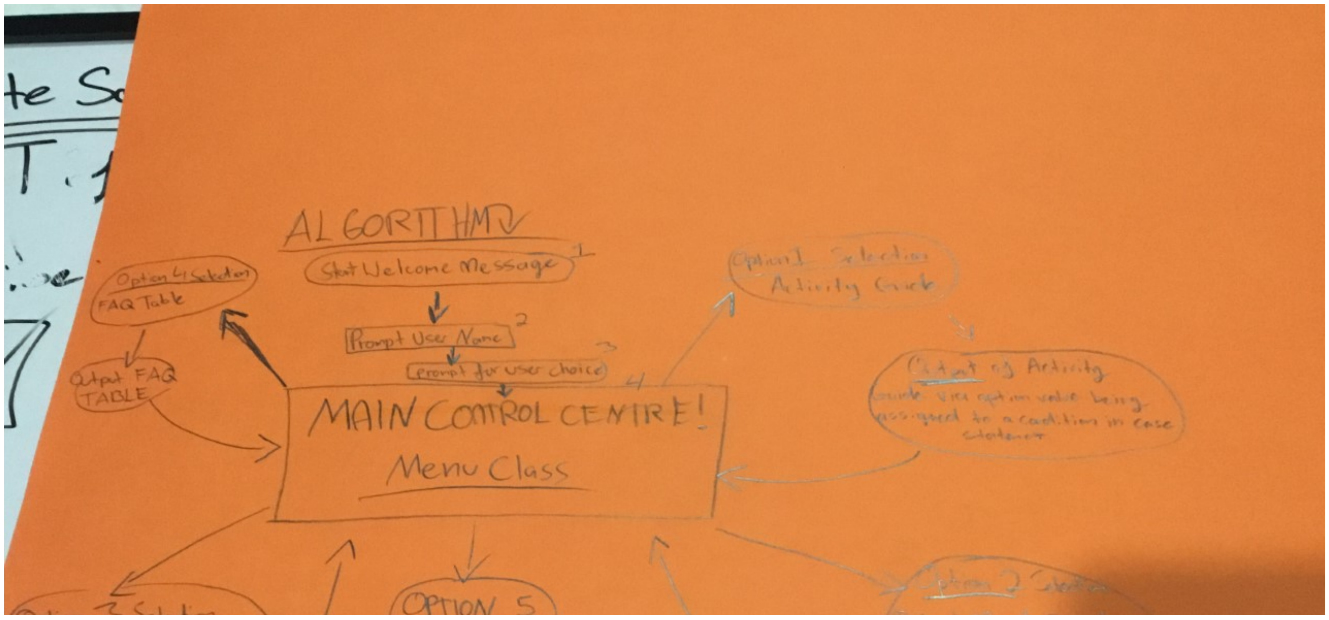
<u>Control Flow Diagram:</u>

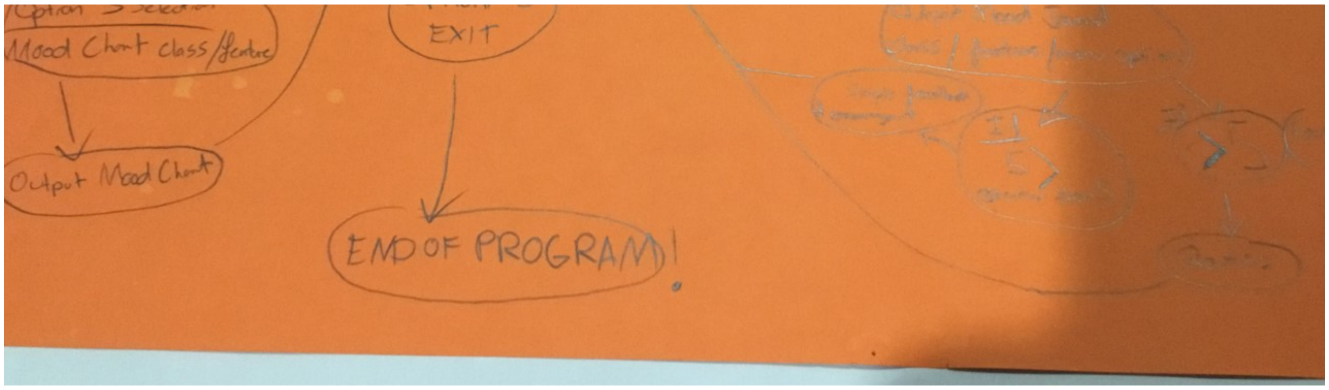
I have included multiple photos of the same algorithm diagram for safe-keeping as the camera quality isn't the best:

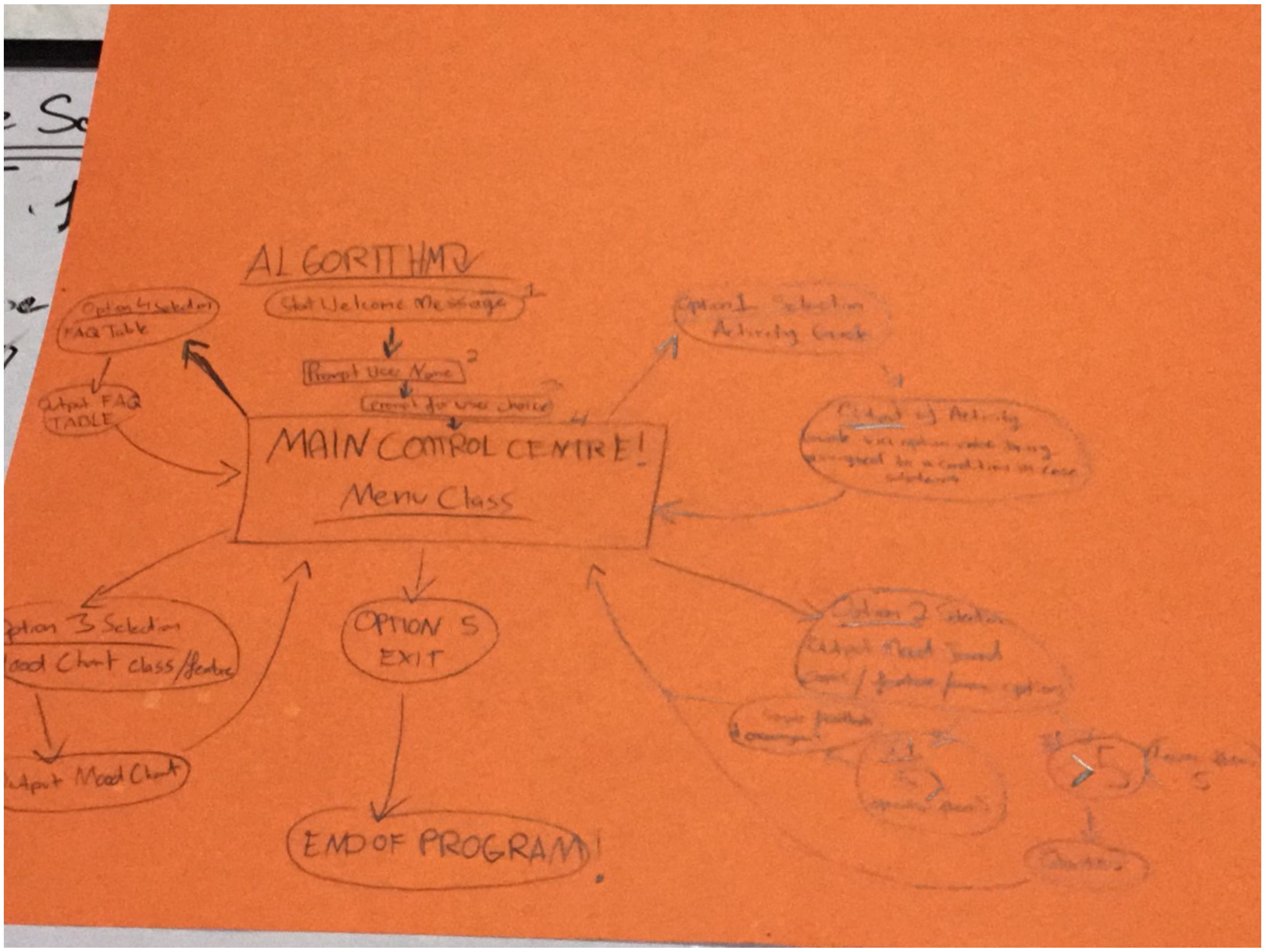


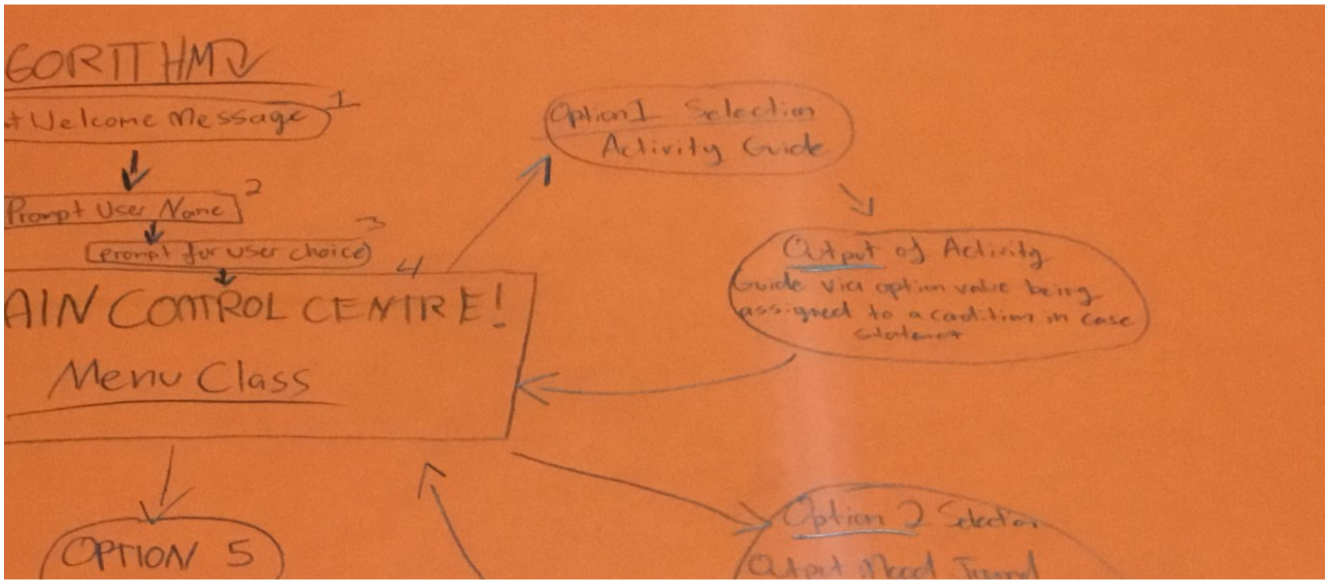

Mood Chart class/feature
Output Mood Chart
EXIT
END OF PROGRAM!
Output Mood Sound
Class / feature / menu option
Simple feedback message
5
> 5
Quotes

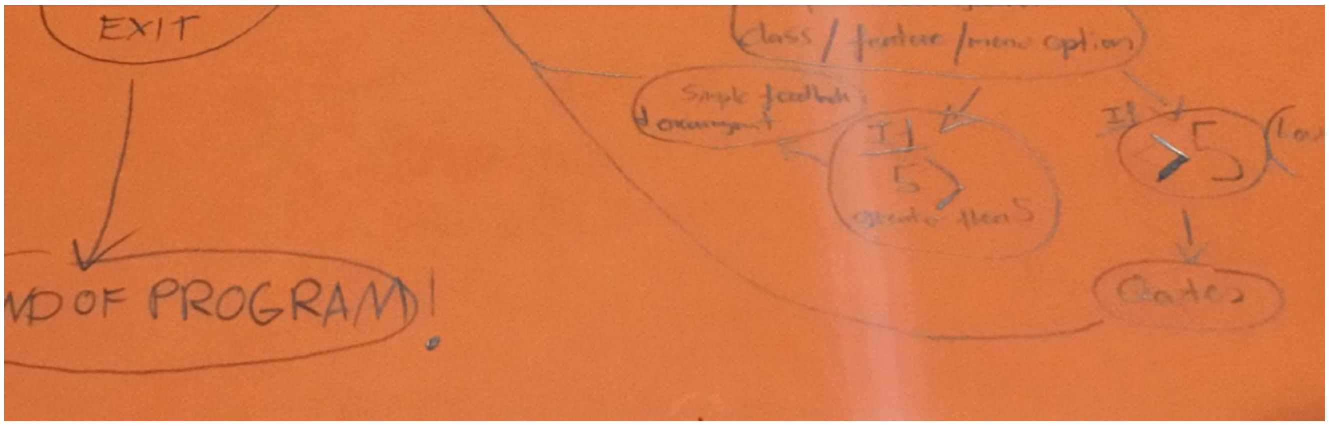

ALGORITHM
Start Welcome Message
1
Prompt User Name
2
prompt for user choice
3
4
MAIN CONTROL CENTRE!
Menu Class
Option 4 Selection
FAQ Table
Output FAQ TABLE
Option 3 Selection
Mood Chart class/feature
OPTION 5
EXIT

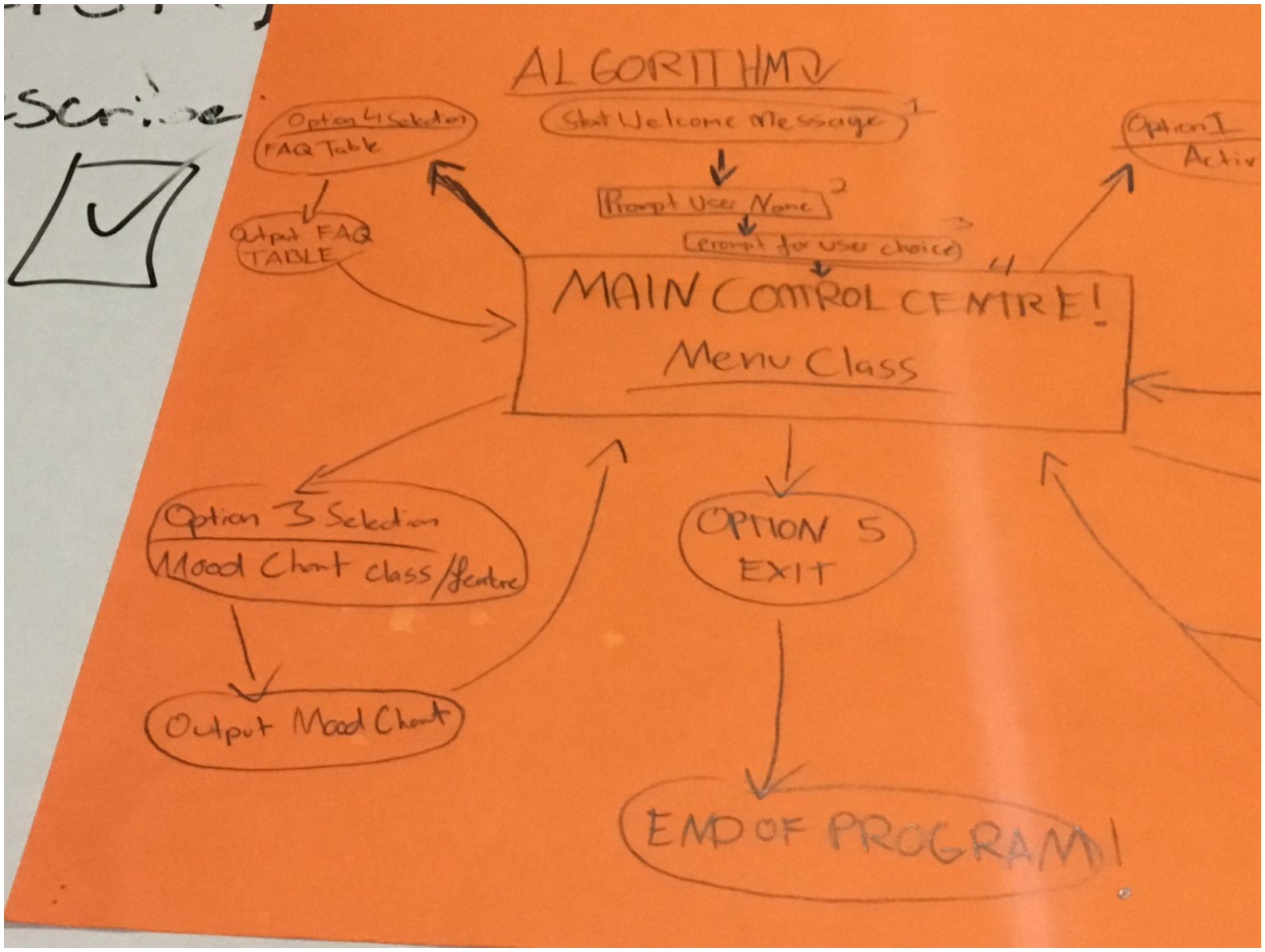

ALGORITHM
Start Welcome Message
1
Prompt User Name
2
prompt for user choice
3
4
MAIN CONTROL CENTRE!
Menu Class
Option 4 Selection
FAQ Table
Output FAQ TABLE
Option 1 Selection
Activity Guide
Output of Activity
Guide via option value being assigned to a condition in case statement
OPTION 5
Option 2 Selection


Option 3 Selection
Mood Chart class/feature
Output Mood Chart
EXIT
END OF PROGRAM!


ALGORITHMS
Start Welcome Message 1
Prompt User Name 2
prompt for user choice 3
4
MAIN CONTROL CENTRE!
Menu Class
Option 4 Selection
FAQ Table
Output FAQ TABLE
Option 1 Selection
Activity Guide
Output of Activity Guide via option value being assigned to a condition in case statement
Option 2 Selection
Output Mood Journal class/feature from option
Option 3 Selection
Mood Chart class/feature
Output Mood Chart
OPTION 5
EXIT
END OF PROGRAM!


ALGORITHMS
Start Welcome Message 1
Prompt User Name 2
prompt for user choice 3
4
MAIN CONTROL CENTRE!
Menu Class
Option 1 Selection
Activity Guide
Output of Activity Guide via option value being assigned to a condition in case statement
Option 2 Selection
Output Mood Journal
OPTION 5


EXIT
class / feature / menu option
Simple feedback / encouragement
If 1 - 5
greater than 5
If > 5
Quotes
END OF PROGRAM!


ALGORITHMS
Start Welcome Message 1
Prompt User Name 2
Prompt for user choice 3
4
Option 4 Selection
FAQ Table
Output FAQ TABLE
Option 1
Activ
MAIN CONTROL CENTRE!
Menu Class
Option 3 Selection
Mood Chart class / feature
Output Mood Chart
OPTION 5
EXIT
END OF PROGRAM!