# DESIGN REPORT

Gym Tracker was designed to help user store data about their workout. Using the SQL data system, it stores many users’ data, help them sort them by data and performance. In addition, it demonstrates their progress using charts drawn in gosu, Finally, it sorts the users data and create a leaderboard.

Program Flowchart

A diagram of a graph

Description automatically generated

1. GymTracker\_functions

* The custom program does not allow $ global variable, therefore I used modules to communicate between the classes as I need to save the data and print them from TextInput class and Home class

Here's an explanation of how the `DataManager` module works:

1. `@selection` is an instance variable within the module. It's an array used to store selected values. This can be thought of as a way to temporarily store some value for later use.

2. `@shared\_data` is another instance variable used to store data that needs to be shared across different parts of the application.

3. `add\_to\_data(value)` is a method that adds the given `value` to the `@shared\_data` array.

4. `get\_data` is a method that returns the contents of the `@shared\_data` array.

5. `delete\_data` is a method that clears the contents of the `@shared\_data` array, effectively resetting it.

6. `add\_selection(value)` is a method that adds the given `value` to the `@selection` array.

7. `delete\_selection` is a method that clears the contents of the `@selection` array.

8. `get\_selection` is a method that retrieves the first value from the `@selection` array. This implies that `@selection` is used to store a single selected value at a time.

By using this module, you can achieve data sharing and management in a more controlled and encapsulated manner compared to using global variables like `$variables`. Global variables can lead to unintended side effects and make it harder to reason about the state of your application.

A screenshot of a computer

Description automatically generated

* Merge sort to sort data

This is a customized merge sort that compares the sum of the 1 and 2 index integer

The provided code contains two functions: `merge\_sort\_2d` and `merge`. These functions together implement a modified version of the merge sort algorithm to sort a 2D array based on the sum of specific columns in each row.

1. `merge\_sort\_2d(arr)` function:

This function is the entry point for the merge sort algorithm. It takes a 2D array `arr` as input and aims to return a sorted version of the input array.

- `num\_elements`: This variable stores the number of rows in the array.

- Base case: If there's only one element or no elements in the array, it returns the array as it is (base case for recursion).

- `middle`: Calculates the midpoint of the array.

- `left\_half`: Splits the input array into the left half, containing elements from index 0 up to (but not including) the middle.

- `right\_half`: Splits the input array into the right half, containing elements from the middle index to the end.

- Recursively calls `merge\_sort\_2d` on the left and right halves.

- Calls the `merge` function to combine and sort the left and right halves.

- Returns the sorted and merged array.

2. `merge(left, right)` function:

This function takes two sorted arrays (`left` and `right`) and merges them while maintaining the sorted order based on a specific criterion (the sum of values in columns 1 and 2).

- `sorted\_arr`: This array will hold the merged and sorted result.

- `left\_index` and `right\_index`: These variables keep track of the current index being considered in the left and right arrays, respectively.

- The `while` loop continues until either the `left\_index` reaches the end of the left array or the `right\_index` reaches the end of the right array.

- `left\_sum` and `right\_sum`: Calculate the sum of values in columns 1 and 2 for the current row in the left and right arrays.

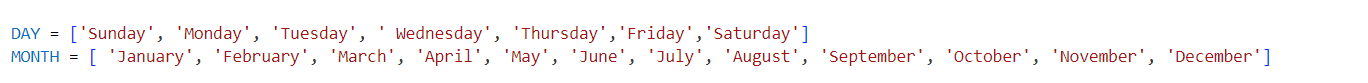
- Compares the sums. If the sum from the left array is less than or equal to the sum from the right array, the row from the left array is added to `sorted\_arr`, and `left\_index` is incremented. Otherwise, the row from the right array is added to `sorted\_arr`, and `right\_index` is incremented.

- This loop continues until one of the arrays is fully processed.

- Finally, any remaining elements from either the left or right array are added to `sorted\_arr`.

In summary, the `merge\_sort\_2d` function recursively divides the input 2D array into smaller halves and then uses the `merge` function to merge and sort these halves based on the sum of values in specific columns. This approach results in a sorted 2D array where rows are ordered based on their column sums.

* A screenshot of a computer code

  Description automatically generated
* Constant for days and time:
* 

1. GymTracker\_users

A screenshot of a computer

Description automatically generated

The menu open up with users and their data

The users names are printed by selecting the users\_name in the table

A diagram with a white rectangle and black arrows

Description automatically generated

When you choose a user, it will assign the user data for them to use it to print later on

A screenshot of a computer program

Description automatically generated

Selection = []  
adding selection

The selection will be the data that connects the users\_id to the right user data

1. GymTracker

A screenshot of a login page

Description automatically generated

A screenshot of a computer

Description automatically generated

A diagram of a graph

Description automatically generated

The work flow of how data is stored and drawn in sql tables, sql is initialized only once when the program is opened or upon require”./”, therefore, for real-time updates on gosu, I programmed it so it inserts data into sql when RETURN key is pressed

A screenshot of a computer program

Description automatically generated

1. The method `sql` is defined. This method seems to be responsible for setting up a connection to an SQLite3 database and performing various database operations.

2. Inside the `begin` block, a connection to the SQLite3 database named "data\_and\_users.db" is established using `SQLite3::Database.new`.

3. The code creates a table named `gym\_users` if it doesn't already exist. This table has columns `users\_id` (an auto-incrementing primary key) and `users\_name` (a non-null text field).

4. Data for gym users is inserted into the `gym\_users` table. Three users with names 'Dang Chi', 'Mackenyu', and 'Sylvain' are inserted.

5. Another table named `gym\_data` is created (if it doesn't exist) to store exercise data. This table has columns `data\_id`, `miles`, `weight`, `time`, and `users\_id`. The `users\_id` column is a foreign key referencing the `users\_id` column in the `gym\_users` table, establishing a relationship between users and their exercise data.

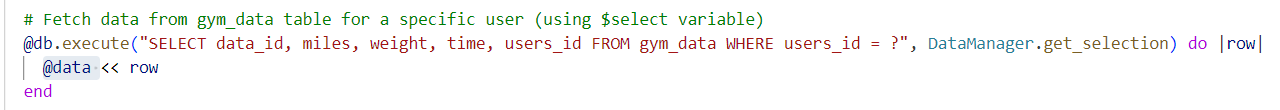
6. The code seems to fetch data from the `gym\_data` table for a specific user. The query uses a placeholder (`?`) and references the `DataManager.get\_selection` method to retrieve the selection criteria.

7. The fetched data is stored in the `@data` array. The exact structure of `@data` isn't provided in the code snippet, but it's assumed to be an array of rows from the fetched result set.

8. The code is wrapped in a `begin` block, which is followed by a `rescue` block. This block catches and handles exceptions of type `SQLite3::Exception` that might occur during database operations. Any caught exception is printed to the console.

9. The `ensure` block is used to close the database connection after all the database operations are done. However, this part of the code is commented out with the comment, "If the whole application is going to exit...". This implies that the code might be part of a larger application where the database connection handling could be managed elsewhere.

* When clicking on a user, it will set your selection so it can find the data from the user



* Here is the textinput to save the data using clone
* A computer code with black text

  Description automatically generated
* A close up of text

  Description automatically generated
* Saving the text input into an array to then be inserted into the sql table and drawn onto the gosu interface
* Greeted by “Today is …” using Time.now.day using module to accurate present the time

A close up of text

Description automatically generated

* You can sort by newest and oldest ( cloned the array and used reverse)
* A screenshot of a computer code

  Description automatically generated
* You can sort by best and worse performance( customized merge sort to compare the sum of the two intergers)
* A screenshot of a computer code

  Description automatically generated
* A status bar will update whether or not you’ve done any exercise today ( using @data)
* A close up of text

  Description automatically generated
* Scrollable gym data records using mouse\_x + 15, add if statements in order for the records to not overflow through the window’s screen

1. Chart screen

A screenshot of a graph

Description automatically generated

* Your progress. The total miles and weight is drawn using each loop and miles weight arrays.

A computer code on a white background

Description automatically generated

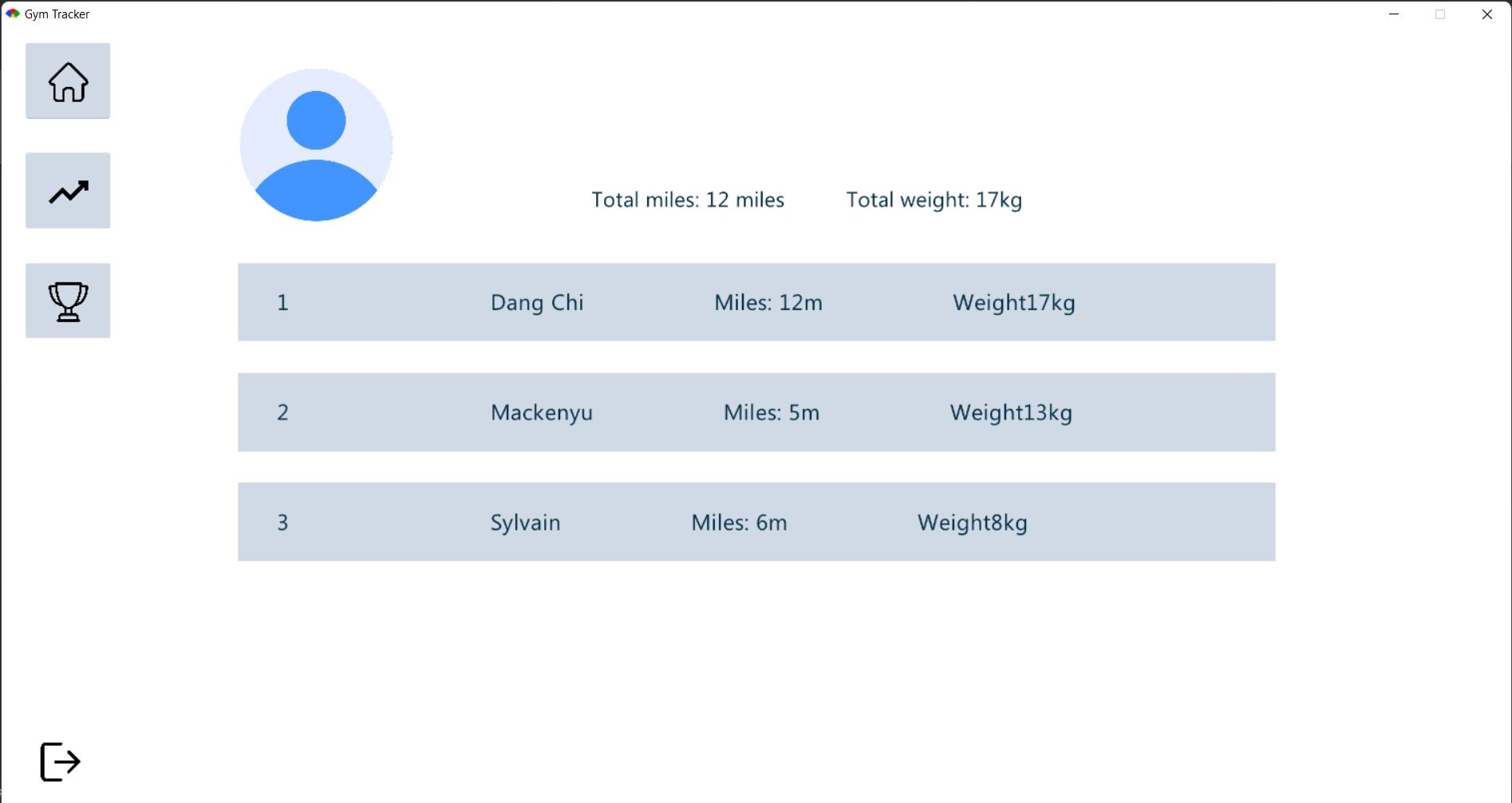
* Drawing the chart: I wanted to draw a line graph, however, it would interfere with eachothers as there are always 1 less line drawn for every dot in the chart, Therefore, I used a loop to draw the circle twice and the line once

A computer code on a white background

Description automatically generated

* You can sort by Total, Miles or Weight ( using Boolean statements and if)

1. Rank screen



* To do the rankings of the accounts, I inner join the tables by users\_id and push the into Array.(@users.size){[]} where @users.size is the amount of users name and {[]} makes every element inside the array an array.
* Then I loop each to replace the arrays in the array with their weight and miles sum for easier comparison

Flowchart:

* A computer screen shot of a computer code

  Description automatically generated
* Then I proceed to sort it using merge sort to produce the leaderboard

Then it is used with merge sort to draw the leader board from most miles and weight to least

A diagram of a flowchart

Description automatically generated