Mobile Device Programming

(COMP3040)

Coursework 1

Name: Tan Zhun Xian

Student ID: 20313854

Lecturer: Doreen Sim Ying Ying

Table of Contents

Section 1: Problem Statements and Motivation of your proposed ideas

In the post-pandemic era, people have become more conscious about health and mental health related issues due to the effects of COVID-19 on the body and also the effects of quarantine on the mind. However, even with more awareness towards health and mental health, people are still clueless on how to manage their health and mental health effectively. So, how can we tackle this problem?

With the prevalence of mobile phones in this digital age, the most effective solution would be to make a mobile app as it gives people a way to monitor their health and mental health more easily. Due to the constraints of time, I have decided to mainly focus on two main aspect which are sleep and exercise. Sleep and exercise are both essential to the health of the body and state of the mind. So, why sleep and exercise specifically?

Without sleep, a person's health will start to deteriorate as their immune system starts to weaken as they continue to forgo sleep. Health issues will soon follow a weakened immune system. This covers the health aspect of the issue. But, what about mental health? Without sleep, a person cannot function properly in their daily lives as they cannot focus and stay awake. This will correlate to a decreased productivity in their lives, and this may lead to feelings of inadequacy and frustration to self and the scorn of others especially by your coworkers and employers. Hallucinations have also been known to follow prolonged lack of sleep and this may cause a person to be in a constant state of fear and paranoia. These issues directly affect a person's mental health and must be handled carefully. So, sleep is particularly important to our health and mental health.

On the other hand, exercise is also important in maintaining a person's health and mental health. Without exercise, the body will become out of shape as the excess fats in the body are not burned through rigorous exercise. This issue is particularly common in the modern era, where the conveniences of life are just a finger-click away. This can cause a multitude of issues such as high blood pressure, diabetes etc. A large body size will also impede the movement of said person and inconvenience their daily lives. This covers the health aspect of exercise. Now, let us study the mental health aspect. In this conforming world, society will often mock what is different from them. This is an issue because humans are social creatures and require human interaction to survive. A large body size is looked down upon and will invite ridicule by our peers. This will have detrimental effects on our mental health. We might feel sad, stressed, depressed and even suicidal. We might even adopt unhealthy practices such as excessive dieting and bulimia to lose weight faster. So, even mental issues will snowball into health issues. So, it is clear to see that exercise is also important to our health and mental health.

Going back to the mobile app, the app solves these issues indirectly by raising awareness of the user towards unhealthy practices in their daily lives, thereby motivating them to promote positive change in their lives. The app achieves this by providing a

platform to record the daily sleep and exercise schedule of the user, allowing them to review their sleep and exercise habits more easily. By allowing the user to see an accurate representation of their habits, they may more easily identify issues such as a lack of sleep or lack of exercise. They may then be motivated to amend these issues, and this will have a positive effect on their health and mental health.

The mobile app allows its users to record information about their sleep schedule such as sleep time and wake time. The sleep duration is calculated automatically from the given information and is then saved into the app.

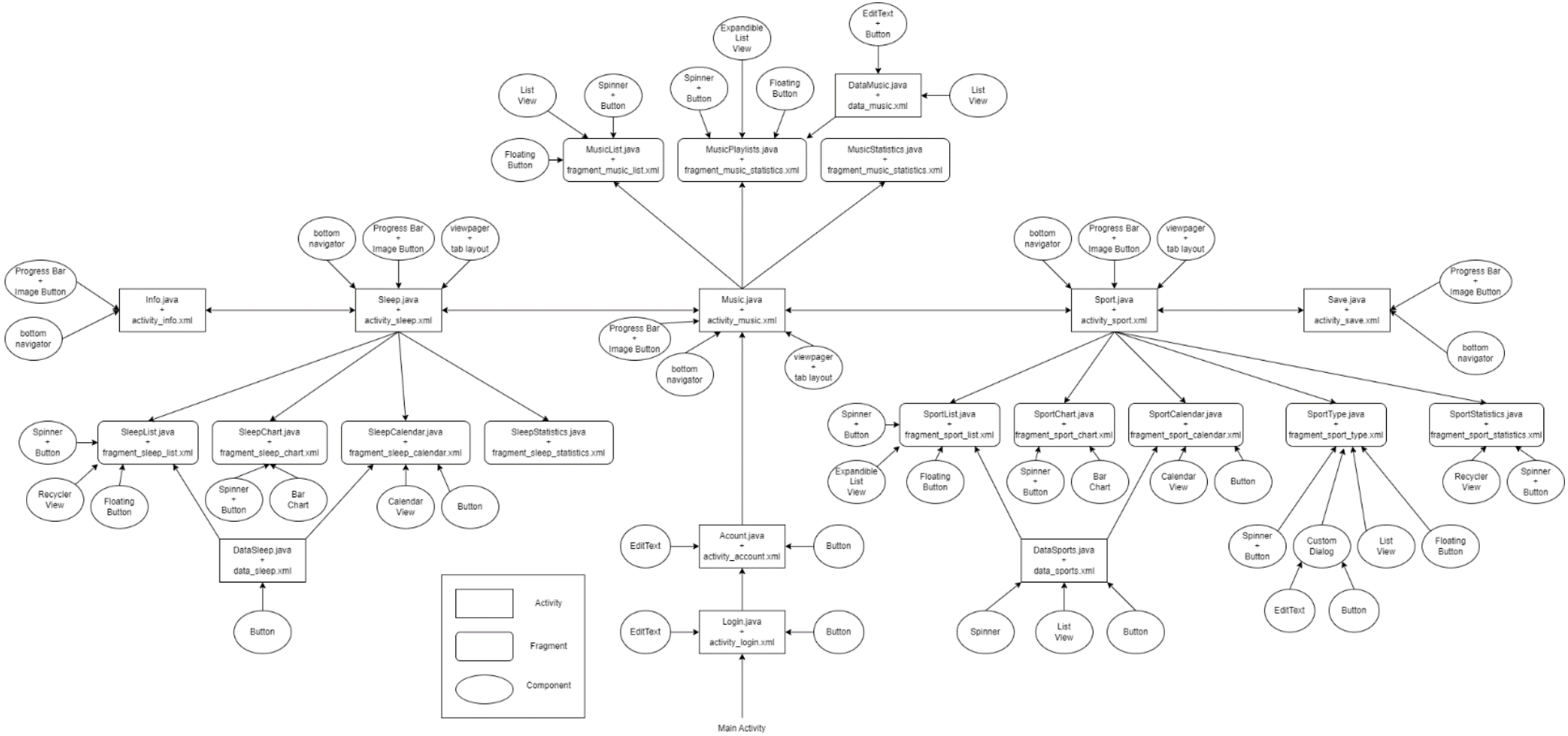
There are more options available for a user to record their exercise schedule. This is because there are a multitude of different options for exercising while sleep generally means the same activity for everyone. So, instead of having the same set of exercises for everyone, the app allows the user to freely add their preferred exercise and delete unused exercises. The app will also allow the user to explicitly set the calories burned per minute for each exercise as each user may burn through different amounts of calories depending on their body weight and effort put into the exercise. Users can specify the type of exercise performed in a specific day and the duration of each exercise. The app will then calculate the calories burned in that day and record the data.

For users to accurately review their sleep and exercise schedules, the app provides options such as lists, charts, and statistics to accurately present and analyze the results. Users can view their sleep and exercise schedules in list or chart form. In the list form, users are also able to sort their data according to different attributes and in both ascending and descending order. For the chart form, users can see their sleep and exercise schedules in temporal order, and this will help the users to spot patterns or trends in their schedules which may be positive and negative and take appropriate action. For statistics, users can see results such as average sleep time, average exercise time and average calories burned. These results may be useful to the user as they may use these results to gauge their progression such as seeing if the average exercise time increases.

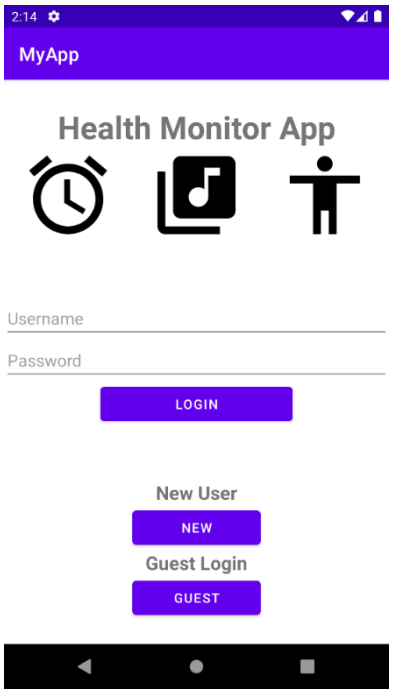
The app also comes equipped with a mp3 player. This addition increases the functions of the app and makes it more than just a health monitoring app. Music has been proven to have a positive effect on mental health as it helps to relieve stress. An improved mental health will also have positive effects on our physical health. Users can freely listen to music before they sleep, during their exercise or any time they want. Users can also import their own songs to the app and create their own playlists of their favorite songs.

So, it is clear to see that my app helps in improving and maintaining health and mental health of its users. My app achieves this by raising awareness of the user towards any unhealthy habits in their daily lives by providing a platform to record and display their daily sleep and exercise schedule.

Section 2: Components and Basic Functionalities + User Interface (UI)



The diagram above shows the main activities, fragments, and components of the app. Some details have been left out as it may overcomplicate the view. Text views are present in all the different activities and fragments so putting them in the diagram would be redundant. Layout information is also absent, but the summary is that the main layout for each activity and fragment is by default a vertical linear layout. However, if the page contains floating or dangling components such as floating action button or bottom navigator, then it is a relative layout. More layout information is not given as most pages have complicated nested layouts. Finally, I decided to combine both Components and Basic Functionalities and User Interface because it makes explaining the involved components and views easier and clearer. Components are bolded and underlined while layouts are placed inside brackets to make them stand out.



This is the launcher activity of the app. [Linear]

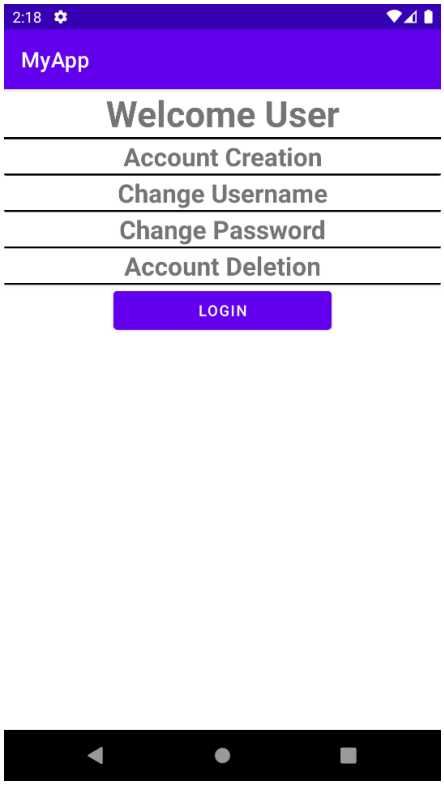
The broadcast receiver notifies the app which user has login so the correct data can be loaded.

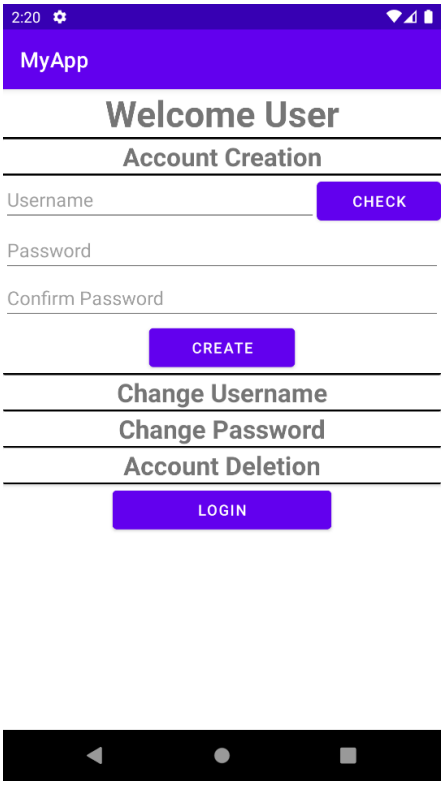
The content provider (database) provides the activity with the list of usernames and passwords to validate the user.

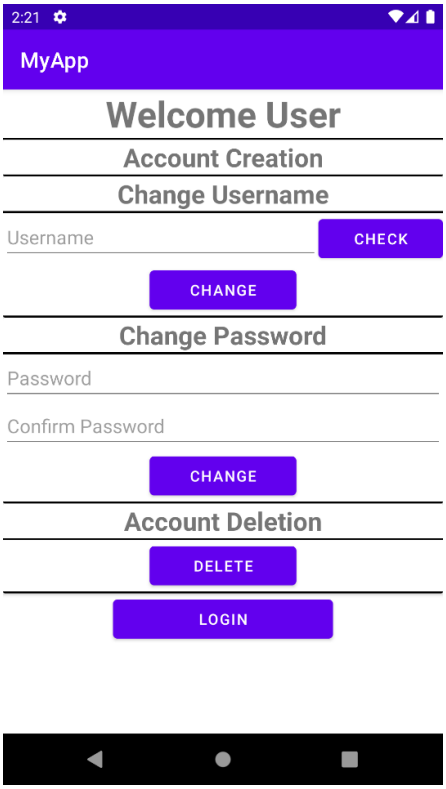
The activity uses a linear layout with edit text to get username and password and buttons to validate input.

Users can login into the app through here. Users with an account can enter their username and password then click the login button to go to next page.

New users can register a new account using the new button while users who want to login as guest can use the guest button.







This is the second activity of the app. [Linear]

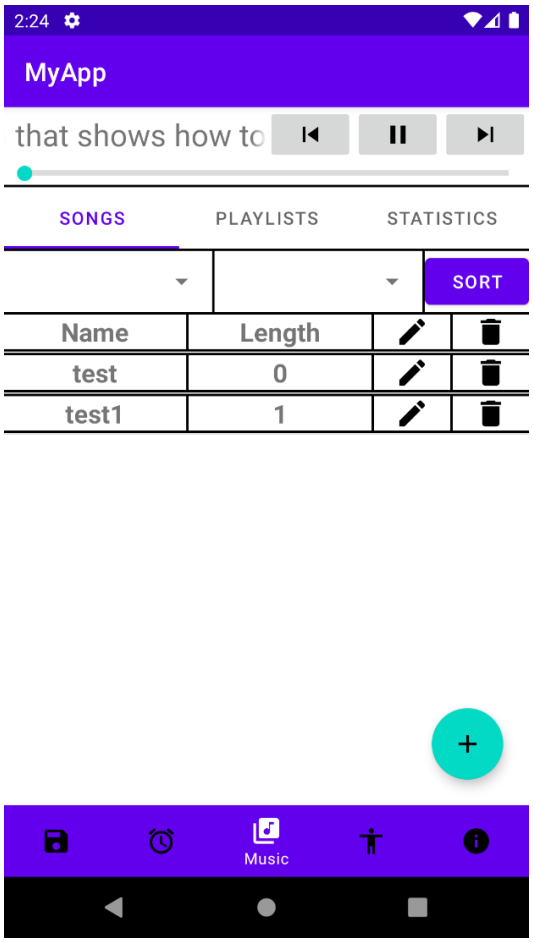
The broadcast receiver notifies the app if there is a change in username or password or account deletion.

The content provider (database) provides the activity with the list of usernames so that existing usernames cannot be chosen.

New and existing users are redirected here. The 4 rows are clickable and will expand when clicked. Account creation is only available to new users. They must enter their desired username and check availability using the check button. Then, they can click the create button.

Existing users can choose to change their username, change their password, or delete their accounts. After everything is done, users can continue into the main app using the login button.

This is the music activity of the app. It has 3 fragments arranged in a tab layout and displayed with a view pager. It has a mp3player on top with image buttons and a seek bar. It also has a bottom navigator to navigate to other activities.



This is the songs fragment of music activity. [Relative]

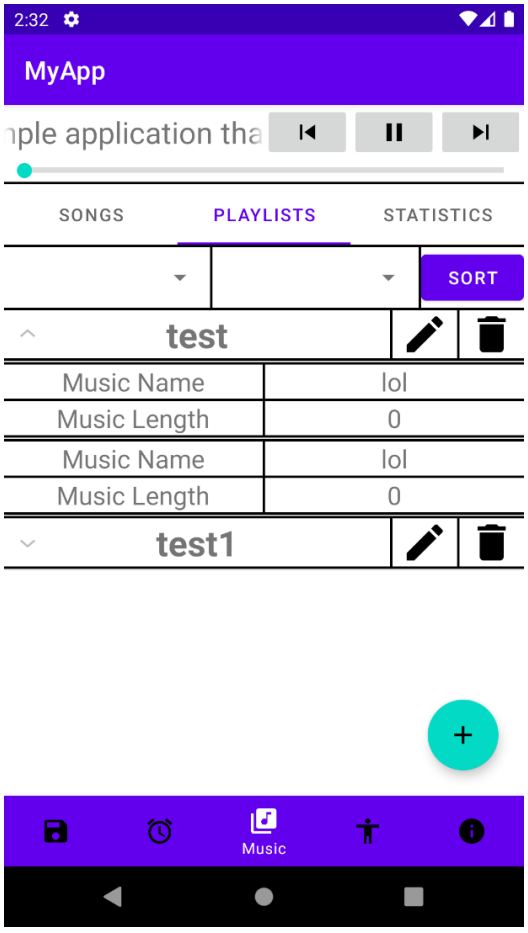
The broadcast receiver notifies the fragment if there is a change in song list data.

The content provider (database) provides the fragment with the song list so that it can be displayed in list form.

There are 2 spinners and a sort button used to sort the song list in a particular order. The first spinner chooses the attribute to be sorted while the second spinner chooses the order of the sorting.

Then, we have the list view holding all the different songs. The first column is the song name. The second column is the song length. The third and fourth are the edit and delete image view buttons.

Then, we have the floating action button used to add new songs to the app.



This is the playlists fragment of music activity. [Relative]

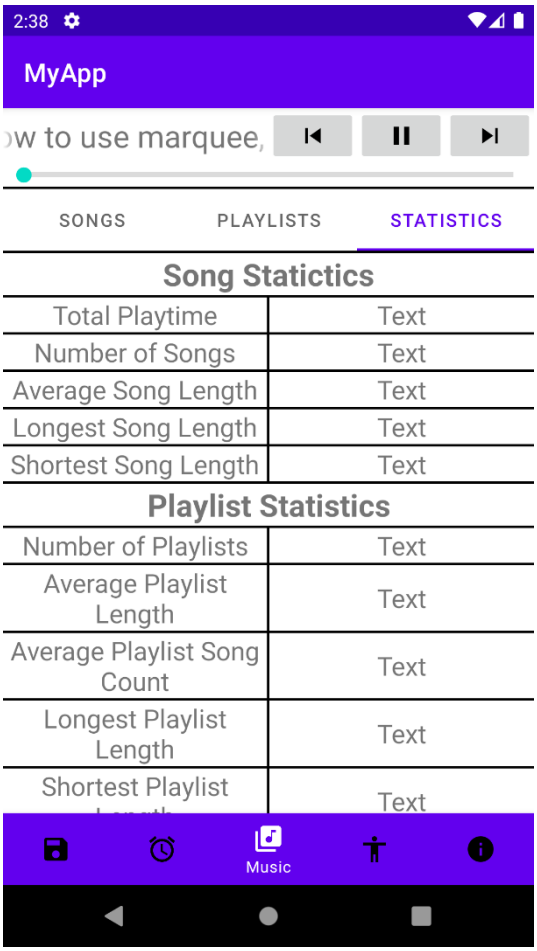
The broadcast receiver notifies the fragment if there is a change in song playlist data.

The content provider (database) provides the fragment with the song playlists so that it can be shown in list form.

There are 2 spinners and a sort button used to sort the song playlists in a particular order. The first spinner chooses the attribute to be sorted while the second spinner chooses the order of the sorting.

Then, we have the expandable list view holding all the different songs of the playlists. Each row expands into a list when clicked. The list shows the song name and length. The edit and delete image view buttons are provided for edit and delete functions.

The floating action button is used to redirect users to another activity to create new playlists.



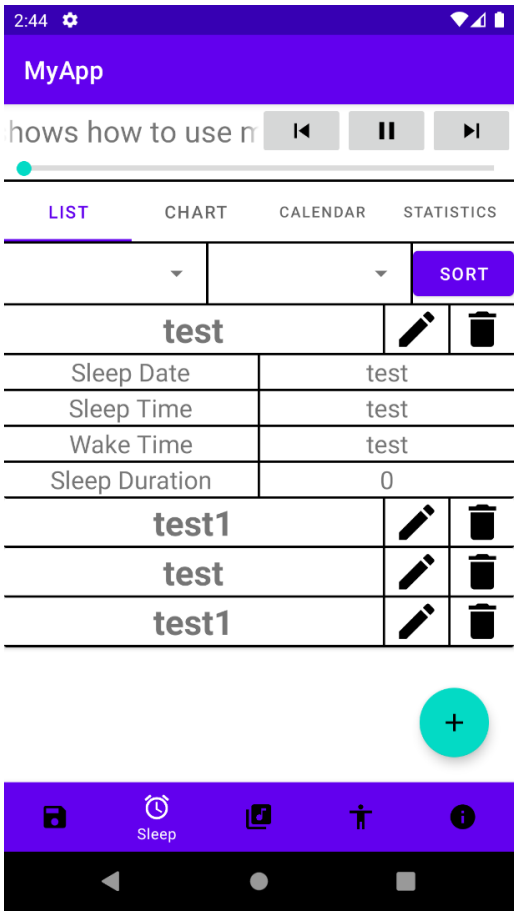
This is the statistics fragment of music activity. [Linear]

The broadcast receiver notifies the fragment if there is a change in song data or song playlist data.

The content provider (database) provides the fragment with the song list and playlists data so that the music statistics can be calculated and displayed.

Users can view the statistics about their songs and playlists here. Statistics such as total playtime, average song length, longest and shortest song length, etc are recorded here.

This is the sleep activity of the app. It has 4 fragments arranged in a tab layout and displayed with a view pager. It has a mp3player on top with image buttons and a seek bar. It also has a bottom navigator to navigate to other activities.



This is the list fragment of sleep activity. [Relative]

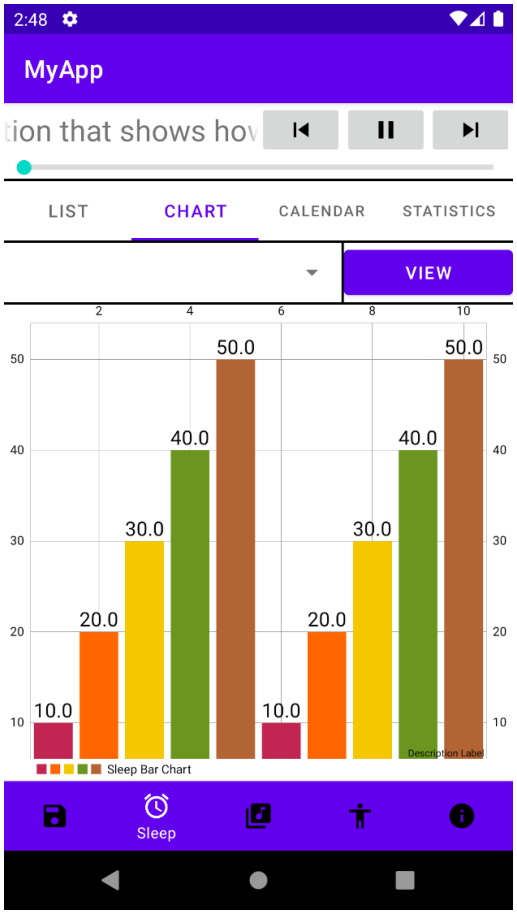
The broadcast receiver notifies the fragment if there is a change in sleep list data.

The content provider (database) provides the fragment with the sleep list data so that it can be shown in list form.

There are 2 spinners and a sort button used to sort the sleep list in a particular order. The first spinner chooses the attribute to be sorted while the second spinner chooses the order of the sorting.

Then, we have a recycler view to show all the different sleep data. Each row is a card view that expands when clicked. Each card view shows the sleep data for a day. The edit and delete image view buttons are provided for edit and delete functions.

The floating action button is used to redirect users to another activity to create new sleep data.



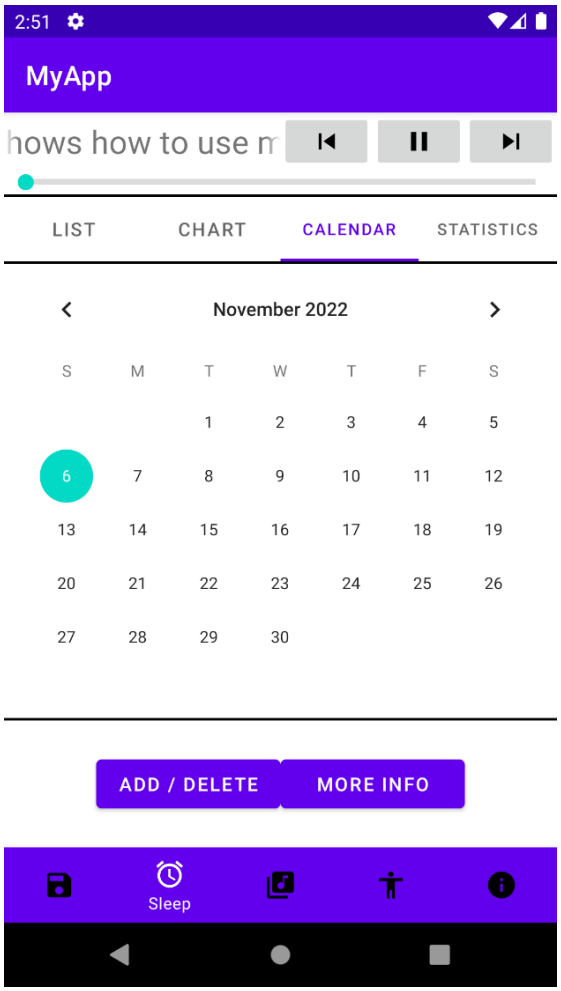
This is the chart fragment of sleep activity. [Linear]

The broadcast receiver notifies the fragment if there is a change in sleep list data.

The content provider (database) provides the fragment with the sleep list data so that it can be displayed in chart form.

There is a spinner and a view button used to view the sleep chart with a particular attribute such as sleep time, wake time or sleep duration.

The bar chart is imported from GitHub and is created by Phil Jay[1]. It is used to display the data in bar chart form for easy visualisation so that a pattern or trend can be observed.



This is the calendar fragment of sleep activity. [Linear]

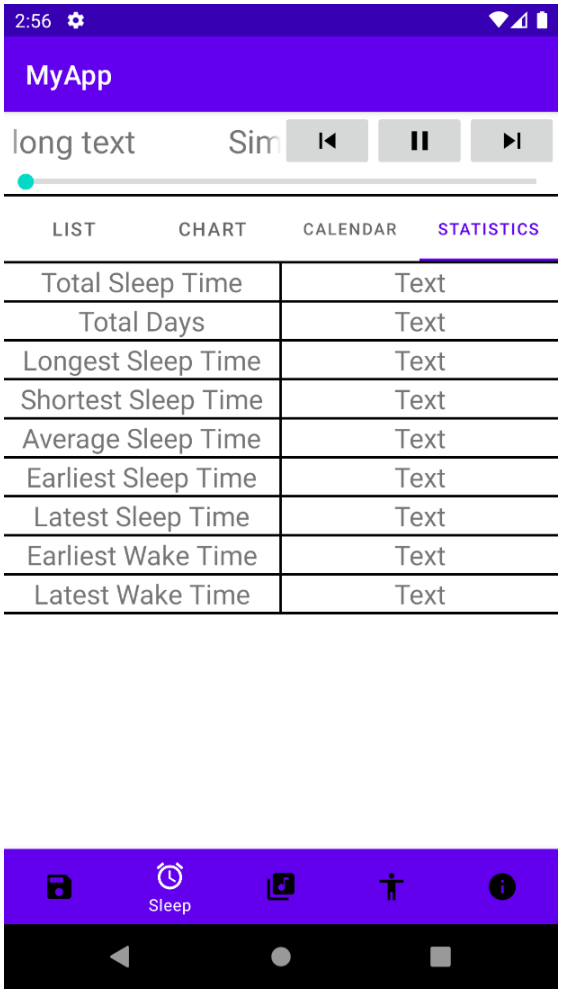
The broadcast receiver notifies the fragment if there is a change in sleep list data.

The content provider (database) provides the fragment with the sleep list data so that it can update the buttons below.

A calendar view is used so that a user can more easily select a date to get more information about that day.

Two buttons are provided below. If the selected day has sleep data, the first button will show delete and the second button will be available. If not, the first button will show add and the second button will be greyed out.

Using the add or more info button redirects the user to another activity to create or modify sleep data while the delete button calls a confirmation dialog.



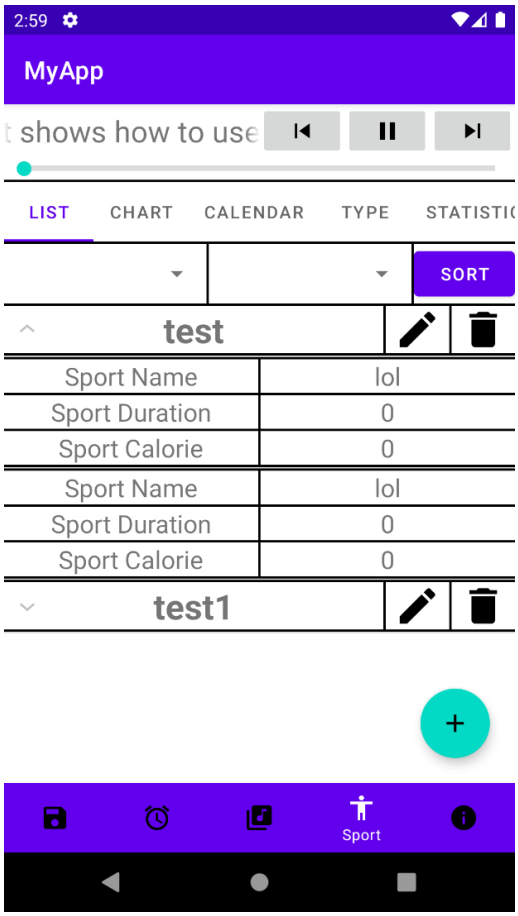
This is the statistics fragment of sleep activity. [Linear]

The broadcast receiver notifies the fragment if there is a change in sleep list data.

The content provider (database) provides the fragment with the sleep list data so that the sleep statistics can be calculated and displayed.

Users can view the statistics about their sleep data here. Statistics such as average sleep time, total sleep time, etc are recorded here.

This is the sport activity of the app. It has 5 fragments arranged in a tab layout and displayed with a view pager. It has a mp3player on top with image buttons and a seek bar. It also has a bottom navigator to navigate to other activities.



This is the list fragment of sleep activity. [Relative]

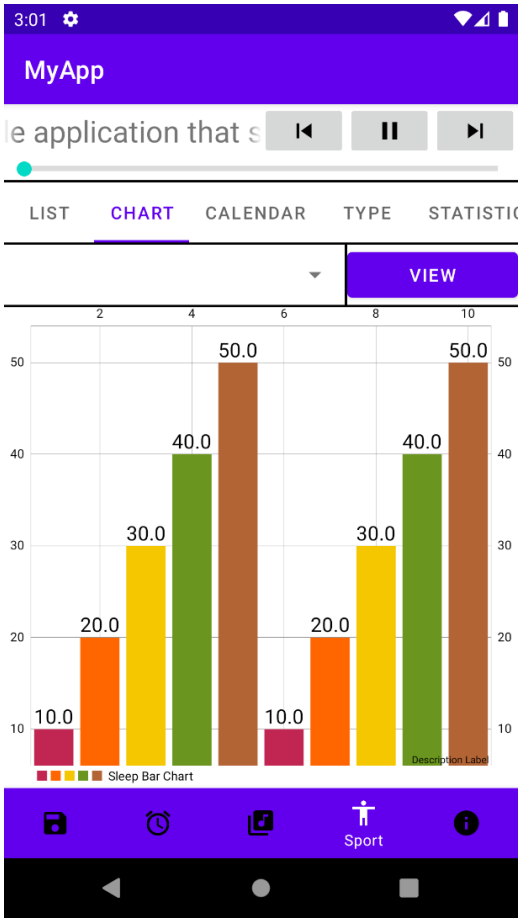
The broadcast receiver notifies the fragment if there is a change in sport list data.

The content provider (database) provides the fragment with the sport list data so that it can be shown in list form.

There are 2 spinners and a sort button used to sort the sport list in a particular order. The first spinner chooses the attribute to be sorted while the second spinner chooses the order of the sorting.

Then, we have a recycler view to show all the different sport data. Each row is a card view that expands when clicked. Each card view shows the sport data for a day. The edit and delete image view buttons are provided for edit and delete functions.

The floating action button is used to redirect users to another activity to create new sport data.



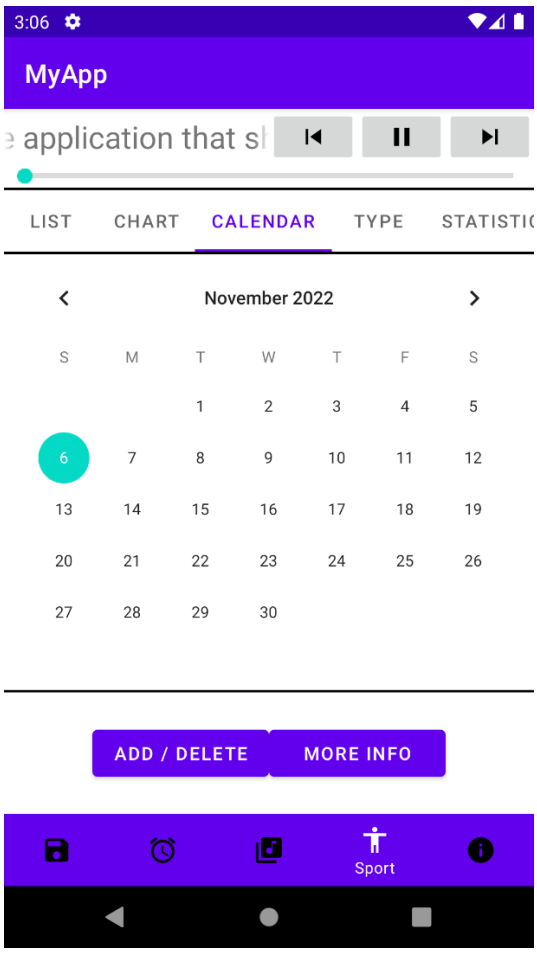
This is the chart fragment of sport activity. [Linear]

The broadcast receiver notifies the fragment if there is a change in sport list data.

The content provider (database) provides the fragment with the sport list data so that it can be displayed in chart form.

There is a spinner and a view button used to view the sport chart with a particular attribute such as sport duration and calories burned.

The bar chart is imported from GitHub and is created by Phil Jay[1]. It is used to display the data in bar chart form for easy visualisation so that a pattern or trend can be observed.



This is the calendar fragment of sport activity. [Linear]

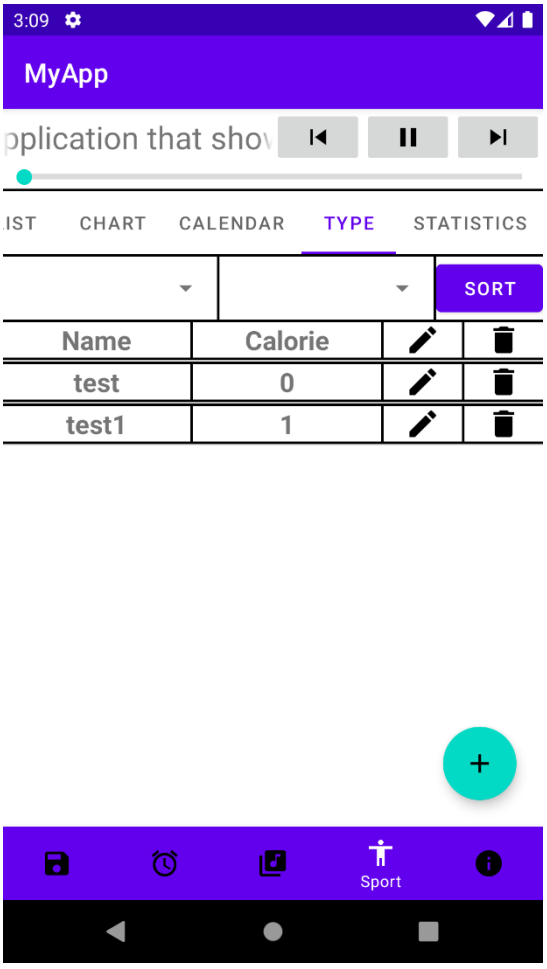
The broadcast receiver notifies the fragment if there is a change in sport list data.

The content provider (database) provides the fragment with the sport list data so that it can update the buttons below.

A calendar view is used so that a user can more easily select a date to get more information about that day.

Two buttons are provided below. If the selected day has sport data, the first button will show delete and the second button will be available. If not, the first button will show add and the second button will be greyed out.

Using the add or more info button redirects the user to another activity to create or modify sport data while the delete button calls a confirmation dialog.



This is the type fragment of sport activity. [Relative]

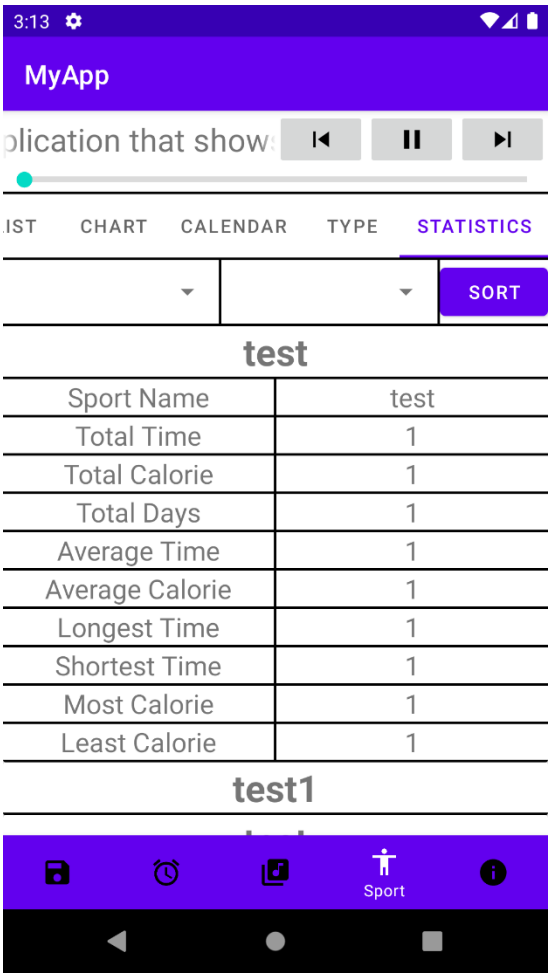
The broadcast receiver notifies the fragment if there is a change in sport type list.

The content provider (database) provides the fragment with the sport type list so that it can be displayed in list form.

There are 2 spinners and a sort button used to sort the sport type list in a particular order. The first spinner chooses the attribute to be sorted while the second spinner chooses the order of the sorting.

Then, we have the list view holding all the different sport types. The first column is the sport name. The second column is the calories burned per minute. The third and fourth are the edit and delete image view buttons.

Then, we have the floating action button used to add new sport types to the app by calling a custom dialog.



This is the statistics fragment of sport activity. [Linear]

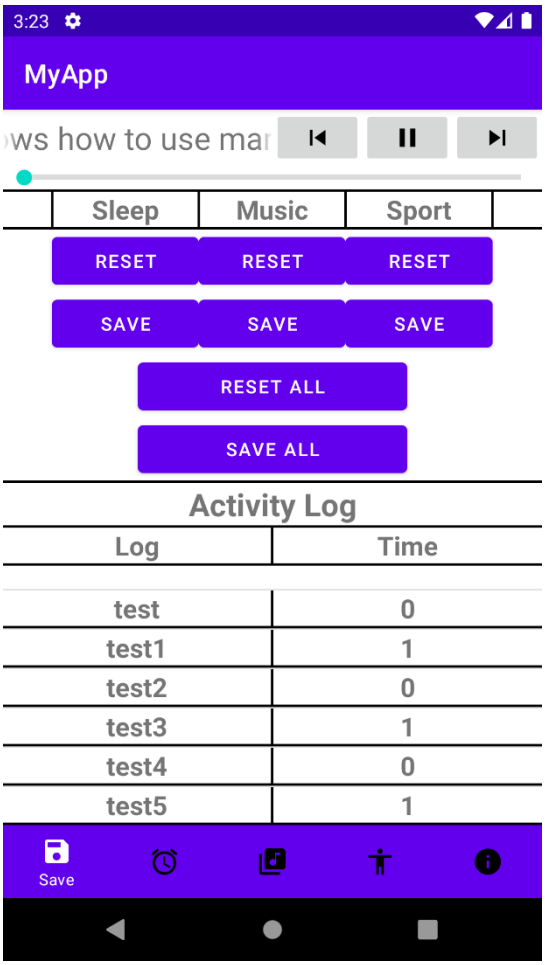
The broadcast receiver notifies the fragment if there is a change in sport list data.

The content provider (database) provides the fragment with the sport list data so that the sport statistics can be calculated and displayed in list form.

There are 2 spinners and a sort button used to sort the sport statistics in a particular order. The first spinner chooses the attribute to be sorted while the second spinner chooses the order of the sorting.

Then, we have a recycler view to show all the different sport type data. Each row is a card view that expands when clicked. Each card view shows the sport data for a sport type.

Users can view the statistics about their sport data here. Statistics such as average calories burned, average sport time, etc. are shown here.



This is the save activity of the app. [Linear]

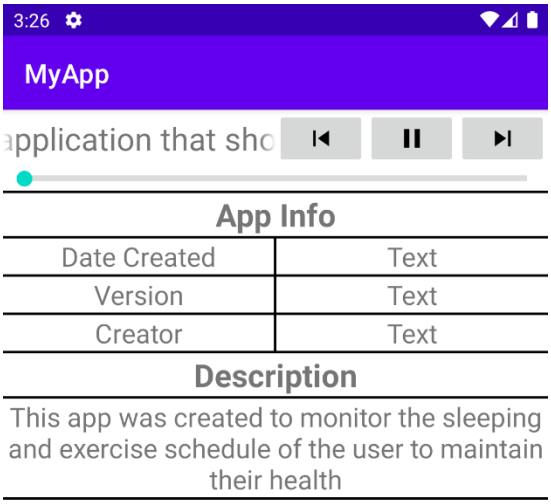
It has a mp3player on top with image buttons and a seek bar. It also has a bottom navigator to navigate to other activities.

The broadcast receiver notifies the activity if there is a change in any data so that it can update the buttons.

The content provider (database) provides the activity with the activity logs of the app so that it can be shown in a list.

The user can reset any changes made to the data using the reset button. Users can also overwrite the old data with new data with the save button. The reset all and save all button are also provided for ease of use. This means that saving is manual in this app and users must remember to save any changes made before closing the app.

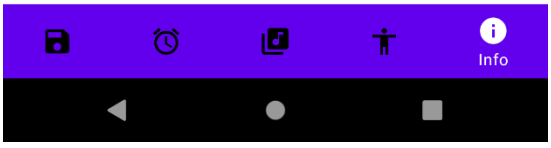
Then, we have the list view holding all the activity logs of the app. The first column is the action performed. The second column is the time the action was performed.

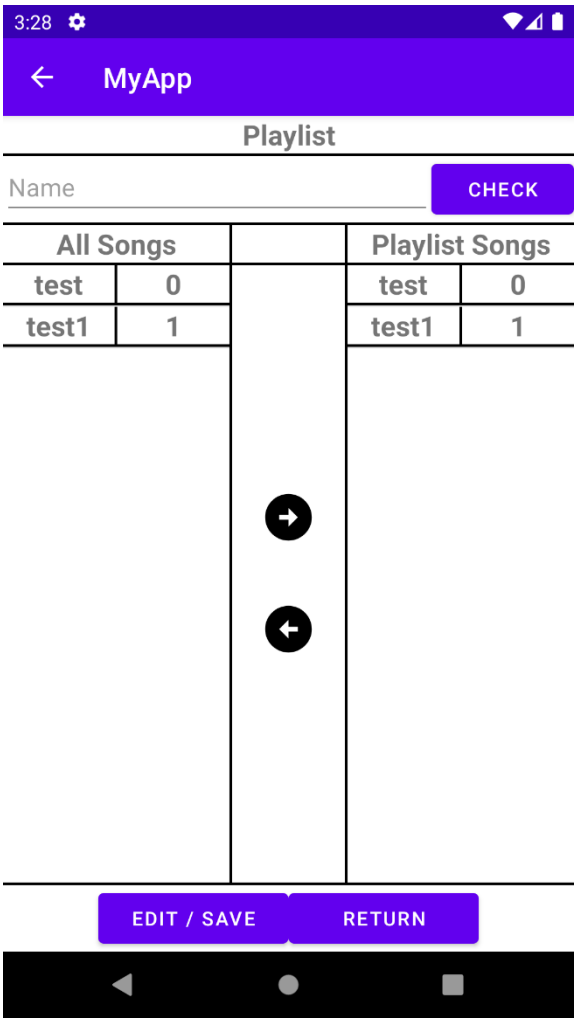


This is the info activity of the app. [Linear]

It has a mp3player on top with image buttons and a seek bar. It also has a bottom navigator to navigate to other activities.

It shows information such as the creator's name, app version and the date the app was created. The activity also has a short description about the app.





This is the Playlist activity. [Linear]

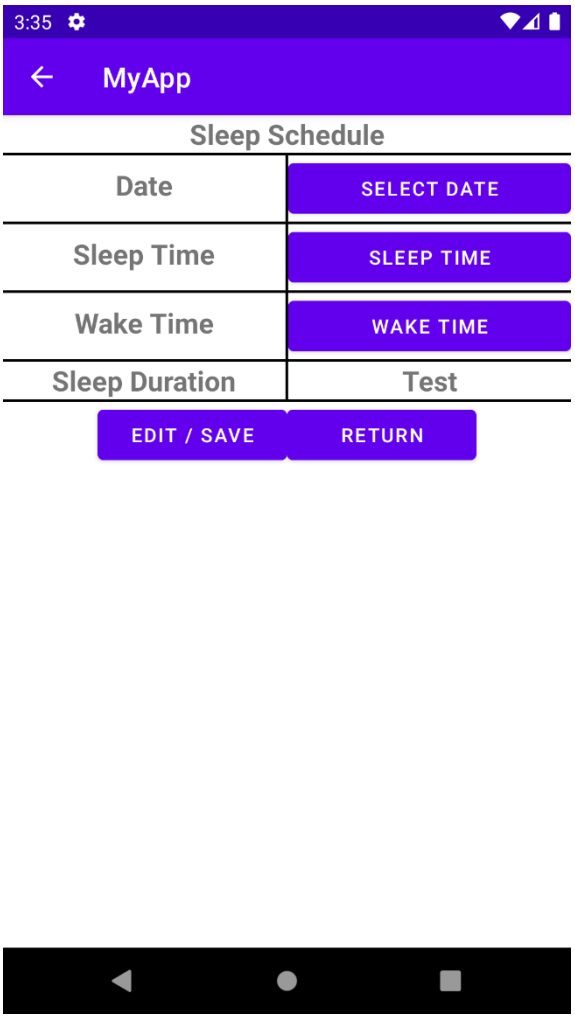
The broadcast receiver notifies the activity if there is a change in song list data.

The content provider (database) provides the activity with the song list data so that it can be displayed in list form.

This page allows the user to create new playlists or edit existing ones. The page has two list views holding all the different songs. The first column is the song name. The second column is the song length.

The left list view holds all the songs available in the app. The right list view are the songs selected to be placed into the new playlist. Songs can be moved using the arrows button below.

The user must first check if the chosen name is available using the edit text and button. Then, the user can start choosing their songs in a new playlist. Finally, the user can save their choice using the save button below.



This is the Sleep Schedule activity. [Linear]

The broadcast receiver notifies the activity if there is a change in sleep list data.

The content provider (database) provides the activity with the sleep list data so that it can be displayed correctly.

This page allows users to record their sleep schedule for a specific day.

When the select date button is clicked, a DatePickerDialog is called, and the player can choose their date. When the sleep time or wake time button is clicked, a TimePickerDialog is called, and the player can choose their sleep and wake time.

The sleep duration is calculated automatically from the sleep and wake time. The user can then save the newly created sleep schedule data using the save button. If the sleep and wake time are invalid, then the user is not able to save the data.

This is the Sport Schedule activity. [Linear]

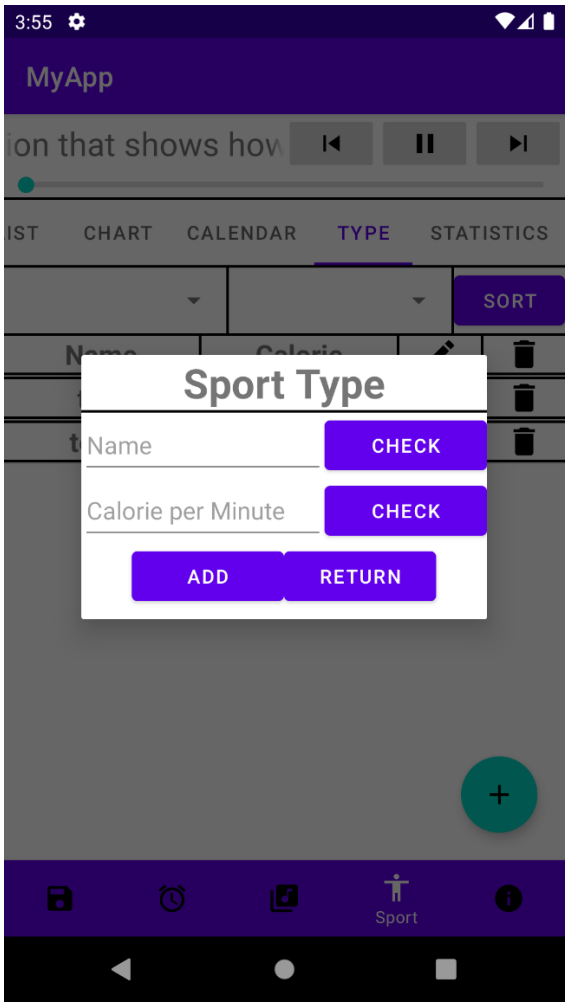
The broadcast receiver notifies the activity if there is a change in sport type data.

The content provider (database) provides the activity with the sport type data so that it can be displayed in the spinner.

This page allows users to select the sports performed in a specific day.

Users must first choose a date using the select date button. Then, users can add their sport type using the first spinner and the sport duration by using the second and third spinner. The total sport duration cannot exceed 24 hours.

The chosen sports and their data are shown in a list view below. Users can delete incorrect data using the delete button. After all additions are made, the user can save the data using the save button.



This is the Sport Type dialog. [Linear]

The broadcast receiver notifies the dialog if there is a change in sport type data.

The content provider (database) provides the dialog with the sport type data list so that the new sport name can be validated.

Before a user can add a new sport type, they must check to see if the given name is valid and if the given calories per minute is also valid using the edit text and buttons.

If both entries are valid, then the new sport can be added to the system using the add button.

Section 3: Use Cases and Use Case Scenarios

Use Case	1
Context	User logins into the app.
Precondition	User is at the login screen.
Scenarios	1. User logins as guest. • User presses the Guest button. • User selects yes on the prompt. 2. User logins as existing user. • User fills in the username and password. • User presses the login button. 3. User logins as new user. • User presses the new button. • User creates new account.
Exceptions	1. User enters wrong username or password in login page. 2. User enters invalid username in account creation.

Use Case	2
Context	User creates new account.
Precondition	User is at the login screen.
Scenarios	1. User is new user. • User presses the new button. • User clicks the account creation row. • User enters valid username and password and passes validation. • User presses create button.
Exceptions	1. User enters invalid username in account creation.

Use Case	3
Context	User changes username.
Precondition	User has login into app as existing user.
Scenarios	1. User is existing user. • User clicks the change username row. • User enters valid username and presses check button. • User presses change button.
Exceptions	1. User enters invalid username in change username.

Use Case	4
Context	User changes password.
Precondition	User has login into app as existing user.
Scenarios	1. User is existing user. • User clicks the change password row. • User enters same password twice. • User presses change button.
Exceptions	1. User enters different password in the two rows in change password.

Use Case	5
Context	User deletes account.
Precondition	User has login into app as existing user.
Scenarios	1. User is existing user. • User clicks the account deletion row. • User presses delete button. • User clicks yes on prompt.
Exceptions	-

Use Case	6
Context	User enters main menu.
Precondition	User has login into app as user.
Scenarios	1. User is existing user. • User clicks login button at the bottom. 2. User is guest user. • User is already at main menu.
Exceptions	-

Use Case	7
Context	User goes to different activity.
Precondition	User has entered main menu.
Scenarios	1. User is in the main menu. • User clicks the image of the activity on the bottom navigator.
Exceptions	-

Use Case	8
Context	User goes to different fragment of the same activity.
Precondition	User has entered the main menu.
Scenarios	1. User likes to tap. • User clicks the tab text on the tab layout. 2. User likes to swipe. • User swipes left or right on screen.
Exceptions	-

Use Case	9
Context	User sorts song list.
Precondition	User has entered the songs fragment of the music activity.
Scenarios	1. User is in the songs fragment of the music activity. • User chooses sort type and sort order in the spinner. • User presses the sort button.
Exceptions	-

Use Case	10
Context	User adds new song to app.
Precondition	User has entered the songs fragment of the music activity.
Scenarios	1. User is in the songs fragment of the music activity. • User presses the floating plus button. • User finds song to be added in the phone. • User adds new song to app.
Exceptions	1. User cannot find song file. 2. User adds invalid file type.

Use Case	11
Context	User changes song name.
Precondition	User has entered the songs fragment of the music activity.
Scenarios	1. User is in the songs fragment of the music activity. • User presses edit button beside song. • User enters new song name. • User presses enter button.
Exceptions	1. User enters existing song name.

Use Case	12
Context	User deletes song.
Precondition	User has entered the songs fragment of the music activity.
Scenarios	1. User is in the songs fragment of the music activity. • User presses delete button beside song. • User presses yes on the prompt.
Exceptions	1. Song file has already been moved or deleted.

Use Case	13
Context	User plays song.
Precondition	User has entered the main menu.
Scenarios	1. User is in the songs fragment of the music activity. • User presses a song in the song list. 2. User is in the playlists fragment of the music activity. • User chooses a playlist. • User presses a song in the playlist. 3. User is anywhere else inside the app. • User presses play button on mp3 player.
Exceptions	1. Mp3 player does not have song loaded in.

Use Case	14
Context	User pauses song.
Precondition	User has entered the main menu.
Scenarios	1. User is anywhere inside the app. • User presses pause button on mp3 player.
Exceptions	1. Mp3 player does not have song loaded in.

Use Case	15
Context	User unpauses song.
Precondition	User has entered the main menu.
Scenarios	1. User is anywhere inside the app. • User presses play button on mp3 player.
Exceptions	1. Mp3 player does not have song loaded in.

Use Case	16
Context	User plays previous song.
Precondition	User has entered the main menu.
Scenarios	1. User is anywhere inside the app. • User presses previous button on mp3 player.
Exceptions	1. Mp3 player does not have song loaded in.

Use Case	17
Context	User plays next song.
Precondition	User has entered the main menu.
Scenarios	1. User is anywhere inside the app. • User presses next button on mp3 player.
Exceptions	1. Mp3 player does not have song loaded in.

Use Case	18
Context	User plays song from a specific point.
Precondition	User has entered the main menu.
Scenarios	1. User is anywhere inside the app. • User drags seek bar to desired position.
Exceptions	1. Mp3 player does not have song loaded in.

Use Case	19
Context	User sorts song playlists.
Precondition	User has entered the playlist fragment of the music activity.
Scenarios	1. User is in the playlist fragment of the music activity. • User chooses sort type and sort order in the spinner. • User presses the sort button.
Exceptions	-

Use Case	20
Context	User edits song playlists.
Precondition	User has entered the playlist fragment of the music activity.
Scenarios	1. User changes playlist songs. • User presses edit button beside song playlist. • User chooses songs in playlist editor. • User presses the save button below. 2. User changes playlist name. • User presses edit button beside song playlist. • User enters valid name into edit text and presses check button. • User presses the save button below.
Exceptions	1. User enters invalid playlist name. 2. User tries to add empty playlist.

Use Case	21
Context	User deletes song playlists.
Precondition	User has entered the playlist fragment of the music activity.
Scenarios	1. User is in the playlist fragment of the music activity. • User presses delete button beside song playlist. • User chooses yes in prompt.
Exceptions	-

Use Case	22
Context	User adds new song playlist.
Precondition	User has entered the playlist fragment of the music activity.
Scenarios	1. User is in the playlist fragment of the music activity. • User presses the floating plus button. • User enters valid name into edit text and presses check button. • User chooses songs in new playlist. • User presses the save button below.
Exceptions	1. User enters invalid playlist name. 2. User tries to add empty playlist.

Use Case	23
Context	User sorts sleep list.
Precondition	User has entered the list fragment of the sleep activity.
Scenarios	1. User is in the list fragment of the sleep activity. • User chooses sort type and sort order in the spinner. • User presses the sort button.
Exceptions	-

Use Case	24
Context	User edits sleep list item.
Precondition	User has entered the list fragment of the sleep activity.
Scenarios	1. User is in the list fragment of the sleep activity. • User presses edit button beside sleep list item. • User edits sleep data. • User presses the save button below.
Exceptions	1. User tries to save invalid range of sleep-wake time.

Use Case	25
Context	User deletes sleep list item.
Precondition	User has entered the list fragment of the sleep activity.
Scenarios	1. User is in the list fragment of the sleep activity. • User presses delete button beside sleep list item. • User chooses yes on the prompt.
Exceptions	-

Use Case	26
Context	User adds new sleep list item.
Precondition	User has entered the list fragment of the sleep activity.
Scenarios	1. User is in the list fragment of the sleep activity. • User presses the floating plus button. • User chooses a date with no sleep data. • User enters new sleep data and presses save button below.
Exceptions	1. User tries to save invalid range of sleep-wake time.

Use Case	27
Context	User views different sleep list chart.
Precondition	User has entered the chart fragment of the sleep activity.
Scenarios	1. User is in the chart fragment of the sleep activity. • User chooses different attribute in the spinner. • User presses view button.
Exceptions	-

Use Case	28
Context	User adds new data to sleep calendar.
Precondition	User has entered the calendar fragment of the sleep activity.
Scenarios	1. User is in the calendar fragment of the sleep activity. • User chooses date with no sleep data on calendar. • User presses add button below. • User enters new sleep data. • User presses save button below.
Exceptions	1. User tries to save invalid range of sleep-wake time.

Use Case	29
Context	User deletes data from sleep calendar.
Precondition	User has entered the calendar fragment of the sleep activity.
Scenarios	1. User is in the calendar fragment of the sleep activity. • User chooses date with sleep data on calendar. • User presses delete button below. • User chooses yes on the prompt.
Exceptions	-

Use Case	30
Context	User edits data from sleep calendar.
Precondition	User has entered the calendar fragment of the sleep activity.
Scenarios	1. User is in the calendar fragment of the sleep activity. • User chooses date with sleep data on calendar. • User presses more info button below. • User edits sleep data. • User presses save button below.
Exceptions	1. User tries to save invalid range of sleep-wake time.

Use Case	31
Context	User sorts sport list.
Precondition	User has entered the list fragment of the sport activity.
Scenarios	1. User is in the list fragment of the sport activity. • User chooses sort type and sort order in the spinner. • User presses the sort button.
Exceptions	-

Use Case	32
Context	User edits sport list item.
Precondition	User has entered the list fragment of the sport activity.
Scenarios	1. User is in the list fragment of the sport activity. • User presses edit button beside sport list item. • User edits sport list item. • User presses the save button below.
Exceptions	1. User tries to save empty sport list item.

Use Case	33
Context	User deletes sport list item.
Precondition	User has entered the list fragment of the sport activity.
Scenarios	1. User is in the list fragment of the sport activity. • User presses delete button beside sport list item. • User chooses yes on the prompt.
Exceptions	-

Use Case	34
Context	User adds new sport list item.
Precondition	User has entered the list fragment of the sport activity.
Scenarios	1. User is in the list fragment of the sport activity. • User presses the floating plus button. • User chooses a date with no sport data. • User enters new sport data. • User presses save button below.
Exceptions	1. User tries to save empty sport list item.

Use Case	35
Context	User views different sport list chart.
Precondition	User has entered the chart fragment of the sport activity.
Scenarios	1. User is in the chart fragment of the sport activity. • User chooses different attribute in the spinner. • User presses view button.
Exceptions	-

Use Case	36
Context	User adds new data to sport calendar.
Precondition	User has entered the calendar fragment of the sport activity.
Scenarios	1. User is in the calendar fragment of the sport activity. • User chooses date with no sport data on calendar. • User presses add button below. • User enters new sport data. • User presses save button below.
Exceptions	1. User tries to save empty sport list item.

Use Case	37
Context	User deletes data from sport calendar.
Precondition	User has entered the calendar fragment of the sport activity.
Scenarios	1. User is in the calendar fragment of the sport activity. • User chooses date with sport data on calendar. • User presses delete button below. • User chooses yes on the prompt.
Exceptions	-

Use Case	38
Context	User edits data from sport calendar.
Precondition	User has entered the calendar fragment of the sport activity.
Scenarios	1. User is in the calendar fragment of the sport activity. • User chooses date with sport data on calendar. • User presses more info button below. • User edits sport data. • User presses save button below.
Exceptions	1. User tries to save empty sport list item.

Use Case	39
Context	User sorts sport type list.
Precondition	User has entered the type fragment of the sport activity.
Scenarios	1. User is in the type fragment of the sport activity. • User chooses sort type and sort order in the spinner. • User presses the sort button.
Exceptions	-

Use Case	40
Context	User edits sport type list item.
Precondition	User has entered the type fragment of the sport activity.
Scenarios	1. User edits sport type list item name. • User presses edit button beside sport type list item. • User enters valid name and presses check button. • User presses the add button below. 2. User edits sport type list item calories per minute value. • User presses edit button beside sport type list item. • User enters valid value and presses check button. • User presses the add button below.
Exceptions	1. User enters invalid sport type list item name. 2. User enters invalid sport type list item calories per minute value.

Use Case	41
Context	User deletes sport type list item.
Precondition	User has entered the type fragment of the sport activity.
Scenarios	1. User is in the type fragment of the sport activity. • User presses delete button beside sport type list item. • User chooses yes on the prompt.
Exceptions	-

Use Case	42
Context	User adds new sport type list item.
Precondition	User has entered the type fragment of the sport activity.
Scenarios	1. User is in the type fragment of the sport activity. • User presses the floating plus button. • User enters valid name and presses check button. • User enters valid value and presses check button. • User presses the add button below.
Exceptions	1. User enters invalid sport type list item name. 2. User enters invalid sport type list item calories per minute value.

Use Case	43
Context	User sorts sport statistics.
Precondition	User has entered the statistics fragment of the sport activity.
Scenarios	1. User is in the statistics fragment of the sport activity. • User chooses sort type and sort order in the spinner. • User presses the sort button.
Exceptions	-

Use Case	44
Context	User resets sleep data.
Precondition	User has entered the save fragment
Scenarios	1. User wants to only reset sleep data • User presses reset button below sleep text. • User clicks yes on prompt. 2. User wants to reset all data. • User presses reset all button. • User clicks yes on prompt.
Exceptions	-

Use Case	45
Context	User resets music data.
Precondition	User has entered the save fragment
Scenarios	1. User wants to only reset music data • User presses reset button below music text. • User clicks yes on prompt. 2. User wants to reset all data. • User presses reset all button. • User clicks yes on prompt.
Exceptions	-

Use Case	46
Context	User resets sport data.
Precondition	User has entered the save fragment
Scenarios	1. User wants to only sport music data • User presses reset button below sport text. • User clicks yes on prompt. 2. User wants to reset all data. • User presses reset all button. • User clicks yes on prompt.
Exceptions	-

Use Case	47
Context	User saves sleep data.
Precondition	User has entered the save fragment
Scenarios	1. User wants to only save sleep data • User presses save button below sleep text. • User clicks yes on prompt. 2. User wants to save all data. • User presses save all button. • User clicks yes on prompt.
Exceptions	-

Use Case	48
Context	User saves music data.
Precondition	User has entered the save fragment
Scenarios	1. User wants to only save music data • User presses save button below music text. • User clicks yes on prompt. 2. User wants to save all data. • User presses save all button. • User clicks yes on prompt.
Exceptions	-

Use Case	49
Context	User saves sport data.
Precondition	User has entered the save fragment
Scenarios	1. User wants to only save sport data • User presses save button below sport text. • User clicks yes on prompt. 2. User wants to save all data. • User presses save all button. • User clicks yes on prompt.
Exceptions	-

Section 4: Key Concerns and Realization of the Challenges of your ideas

Part 1: Assumptions

The app assumes that its users have adequate deductive reasoning skills as no tutorial is provided. Users are expected to fumble their way around the app themselves and deduce the functions of each page through the words and images shown. So, the app is designed to be as intuitive as possible.

The app also assumes that the user is reasonable about the usage of the app. This means that the user will not input invalid values into the app such as a wake time earlier than their sleep time and an exercise schedule more than 24 hours long. The app provides some safeguards against these invalid attempts, but it is generally better if the user is reasonable.

The app also assumes a small amount of data. The app is not designed for large datasets numbering in the millions. A large dataset may cause lag in the app and may affect user experience. It is designed for the use of a small number of users.

The app also assumes that the user will give permission to read and write external storage data as it requires this permission to import new song files into the app and change song names. If no permission is given, the add song and change song name function may not function properly.

Part 2: Key Concerns and Challenges

The main challenge of this app is time. The functions detailed in this app are numerous and are not simple to implement. With enough time, the app is expected to be able to be completed. However, the need to balance my time with other subjects may negatively impact the production of this app. So, some functions may be less sophisticated than originally planned and the look of the app may be less appealing as aesthetics may be sacrificed for functionality.

Other than that, some of the functions detailed in the app are also quite complex and may be quite difficult to implement. This is an issue because I might not be able to implement the functions properly due to a lack of programming knowledge and experience. So, some functions may be repurposed or may be missing entirely.

Furthermore, the app is also planned to the function of modifying files from the external storage of the phone. This is to accommodate the function of adding songs and editing song names of the app. The methods involved are more technical and may be more difficult to implement.

Finally, the app also must create and maintain a database. This is to save the user sleep and exercise schedule data. The data must be saved and loaded every time the user uses the app. This function may prove difficult to implement as we must use an entirely different system entirely, namely MySQL. So, the function may be exceedingly difficult to implement. Text files may have to be used as the database if implementation of the MySQL database fails.

Part 3: Resources and Implementation

The app requires GitHub resources by PhilJay[1] to implement the bar chart as the base functions in Android Studio do not provide bar chart functionality. The bar chart is used to display the user sleep and exercise schedule overtime so that a habit or trend may be more easily discerned. The bar chart is also scrollable to accommodate the large amount of data expected.

The app also requires access to a database to store the results. The database system chosen is MySQL. It is used to store user sleep and exercise schedule data. The data is loaded and saved every time the user uses the app. Different users have different data files and the data files to be loaded are decided during the login process. Guest users have the default dataset and do not have the option to save or load their data. The data can be reset manually by the user to the last save point by pressing the reset button. The save function of the app is also manual so users should always remember to save the changes to their data by pressing the save button.

The calendars of the app are implemented using calendar view and the data is managed using buttons below the calendars.

The app also has an mp3 player on all 5 main activities. The mp3 player has a text view to show current song name, image buttons to pause/play song and move to previous/next song, and a seek bar to see song progress and replay songs from a certain point.

Users can navigate to different activities by using the bottom navigator. There are 5 main activities in the app. They must click the image on the bottom navigator to go to the activity.

The 3 main activities processing data (Music, Sleep, Sport) have fragment adapters managing different fragments for each activity. The fragments are arranged in a tab layout and are displayed by view pagers. They can be accessed by clicking the tab item on top or by swiping the screen left or right. Music activity has 3 fragments, Sleep activity has 4 fragments and Sport activity has 5 fragments.

For text input by the user, an edit text view and a button are generally used for this purpose. The edit text view is used to get input from the user while the button is used to verify if the input is valid. As an example, valid names mean that the input name has not already been taken while valid numbers only contain numeric data and nothing else. Passwords are validated using two different edit text view to ensure that a faulty password does not go through.

All data in the app is available in list form for the view of the user. There are 3 main types of lists used which are list view, recycler view and expandable list view.

List view is used when the data to be shown is few and can be displayed in a few short lines. The data shown also has no special functions such as expanding and collapsing when pressed. List view is used to display song list and sport type list. This is because both list data are short and do have any special functions.

Recycler view is used when the data to be shown is numerous. It requires expanding and collapsing function when clicked to accommodate the data and it accomplishes this using a collapsible card view. The list data is homogeneous as they share the same template file. Recycler view is used to display sleep list and sport statistics list. This is because both list data are long, and they all share the same template.

Expandable list view is used when the data to be shown is variable. It also expands and collapses when pressed. The list data share the same template, yet the number of list data may be different for each list. Expandable list view is used for music playlists and sport list. This is because both lists may vary in size, yet the list data share the same template.

The app also provides sort functionality for the different lists. This function is generally implemented using spinners and a button. The first spinner chooses the sort type while the second spinner choose the sort order. The button is used to validate the choice.

For the add/edit functions of the app, the user is generally taken to a new activity to edit their data before adding it to the app. The two exceptions are add new song, where the user is redirected to an external file manager to find the song file, and add/edit sport type, where a dialog is called as we only process two lines of data which cannot justify the creation of a new activity. The add function is usually called with the floating action button or a normal add button. The edit function is usually called using the edit image button or normal edit button.

For the deletion function of the app, a dialog box is always called to validate the choice. This is to ensure that users do not accidentally delete their data by a mis click. The delete function is usually called with the delete text button or normal delete button.

Part 4: Motivation

My motivation to make the app is so that I can make a difference in the world. I want to be able to help even a single person in improving and maintaining their health and mental health. I understand how the average user cannot easily notice a pattern by looking at raw data. So, I decided to implement views such as lists and charts so that the users can more accurately visualize the data and notice a pattern or trend.

Other than that, I am also motivated to make the app because I can improve my programming skills. The proposed ideas in the app are quite difficult and may prove a challenge to me. However, if I manage to complete the app, I will gain valuable experience and knowledge, and this may help me in my professional life.

Furthermore, this app would also help in enriching my CV. The app is quite difficult and if completed can be proof of my programming dedication and skills. Its inclusion into my CV may help in increasing my desirability to employers as it helps me to stand out from the other aspiring programmers.

Finally, I am also motivated to complete the app to get good marks in the subject. The app is quite advanced and if completed can help to secure a good mark in the module and help advance my studies.

Section 5: References

1. https://github.com/PhilJay/MPAndroidChart
2. https://developer.android.com/reference/android/widget/ListView
3. https://developer.android.com/reference/android/widget/ExpandableListView
4. https://developer.android.com/develop/ui/views/layout/recyclerview
5. https://stackoverflow.com/questions/35500322/expand-and-collapse-cardview