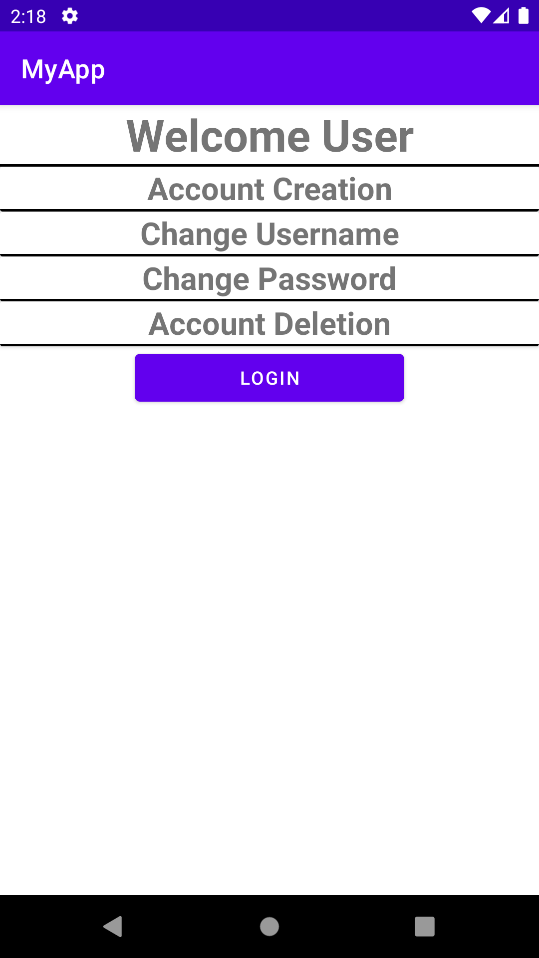
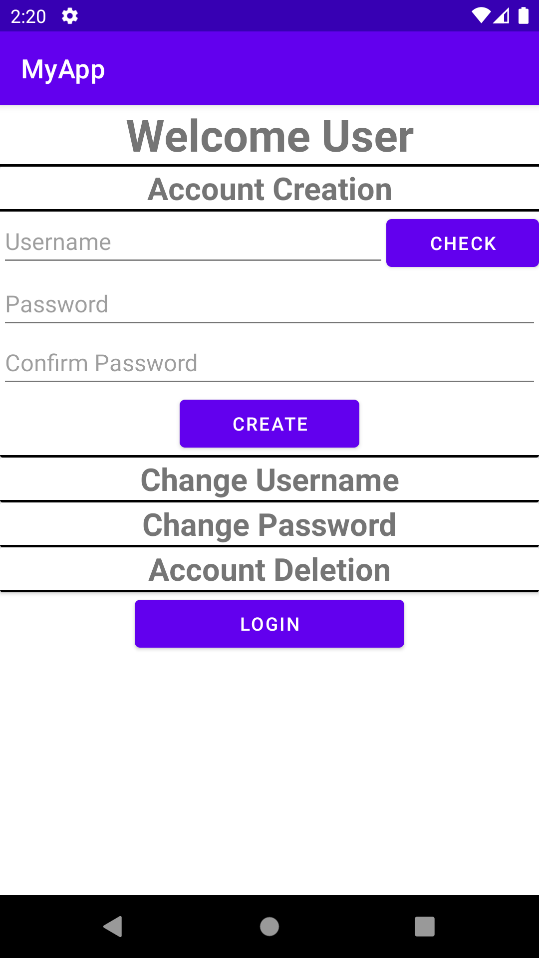
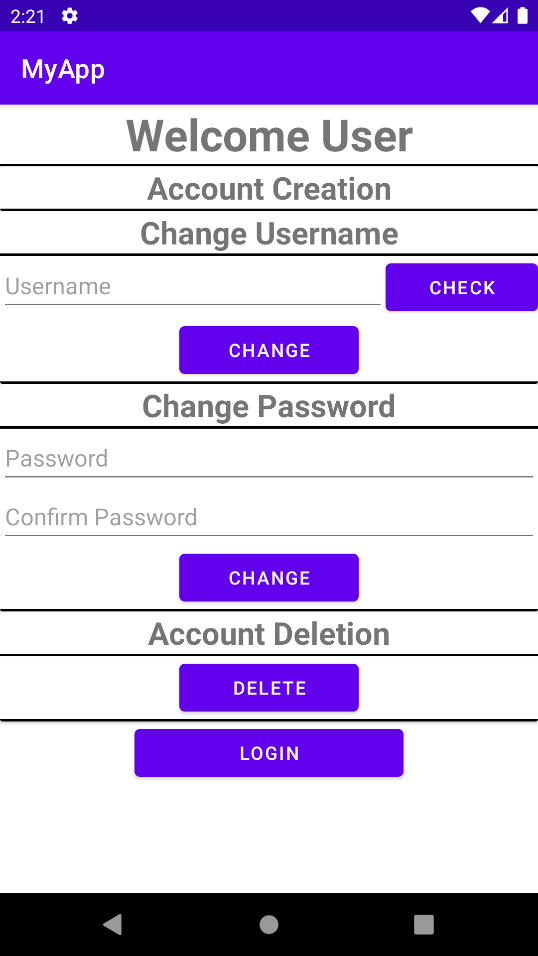
This is the launcher **activity** of the app. [**Linear**]

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**]

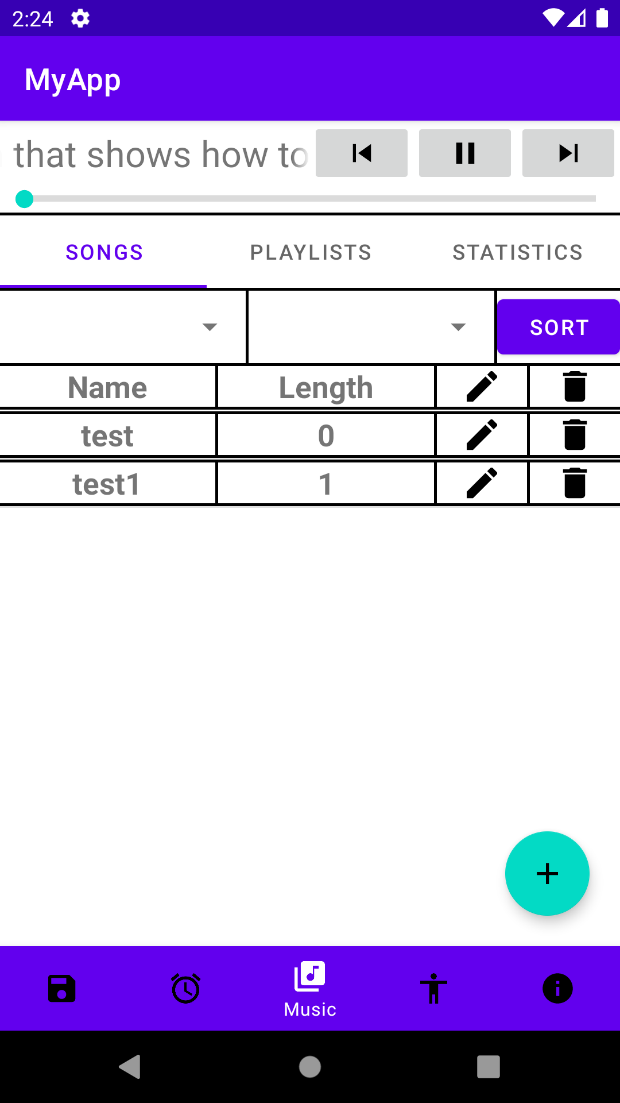
The **broadcast receiver** notifies the app if there is a change in username or password or account deletion.

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 first **fragment** of music activity. [**Relative**]

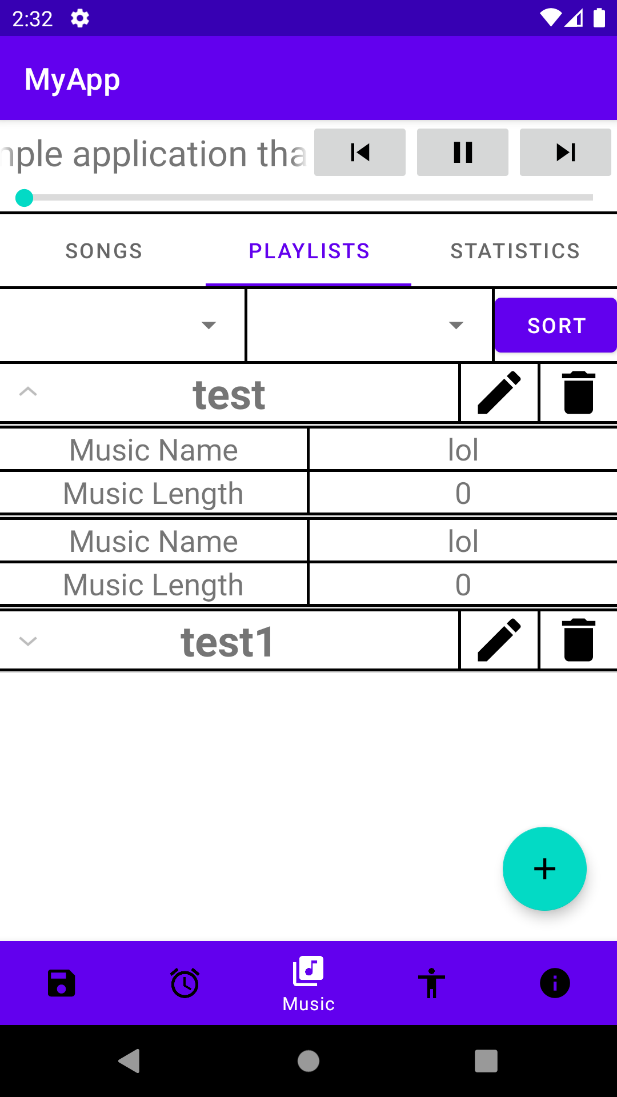
The **broadcast receiver** notifies the fragment if there is a change in song list data.

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 button** used to add new songs to the app.



This is the second **fragment** of music activity. [**Relative**]

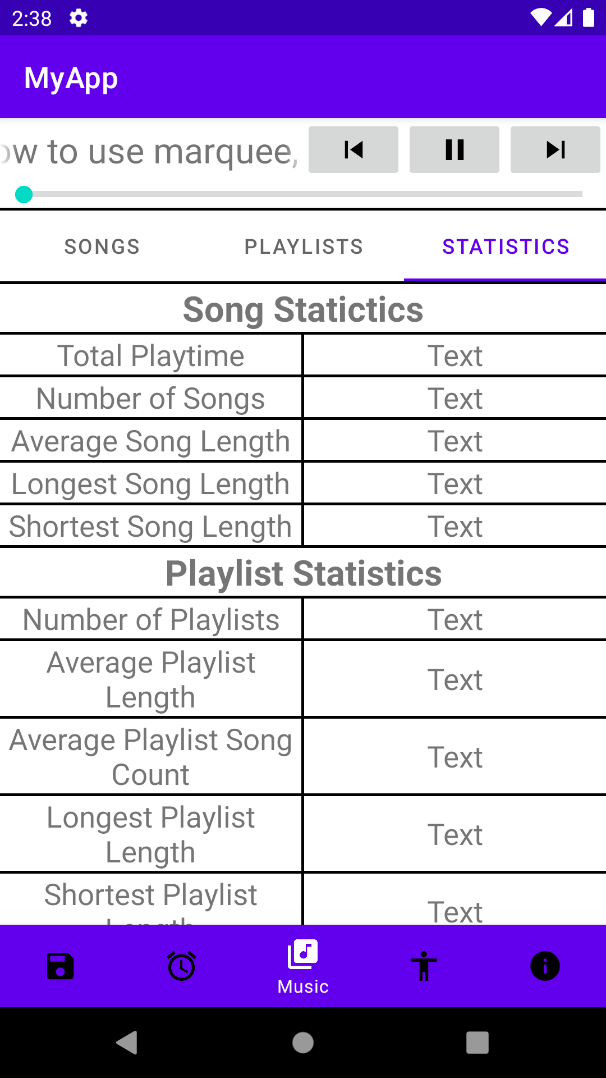
The **broadcast receiver** notifies the fragment if there is a change in song playlist data.

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 button** is used to redirect users to another activity to create new playlists.

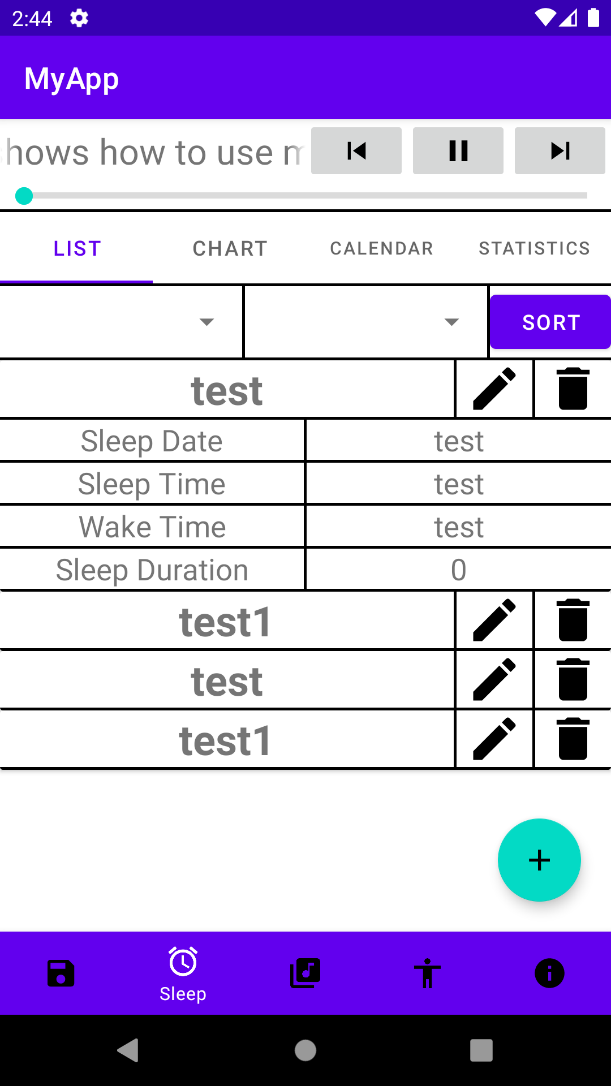
This is the third **fragment** of music activity. [**Linear**]

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 first **fragment** of sleep activity. [**Relative**]

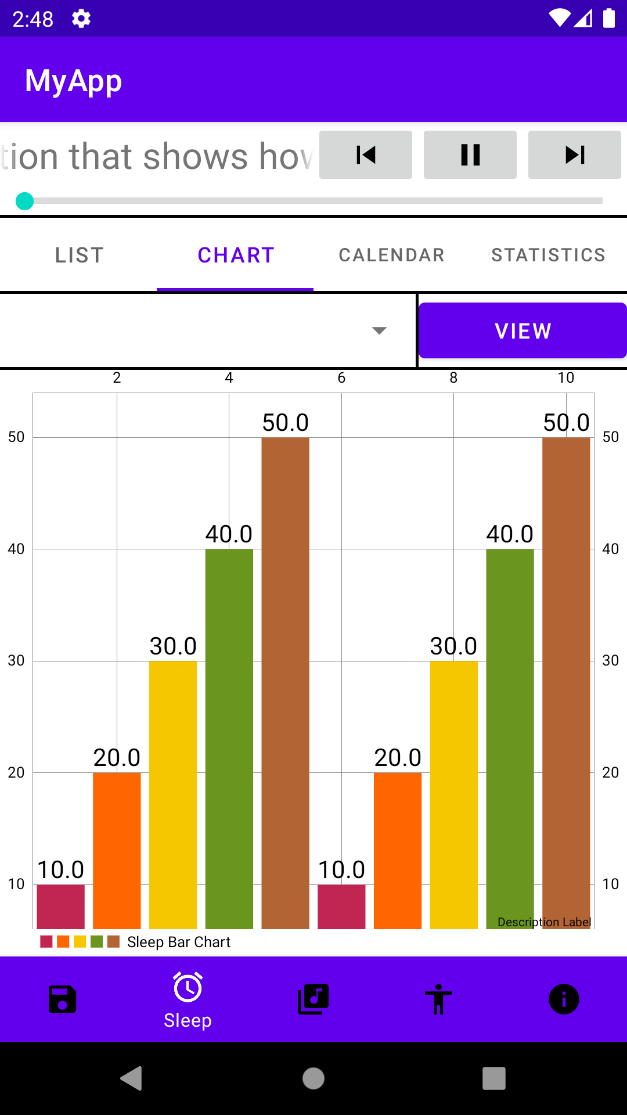
The **broadcast receiver** notifies the fragment if there is a change in sleep list data.

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 button** is used to redirect users to another activity to create new sleep data.



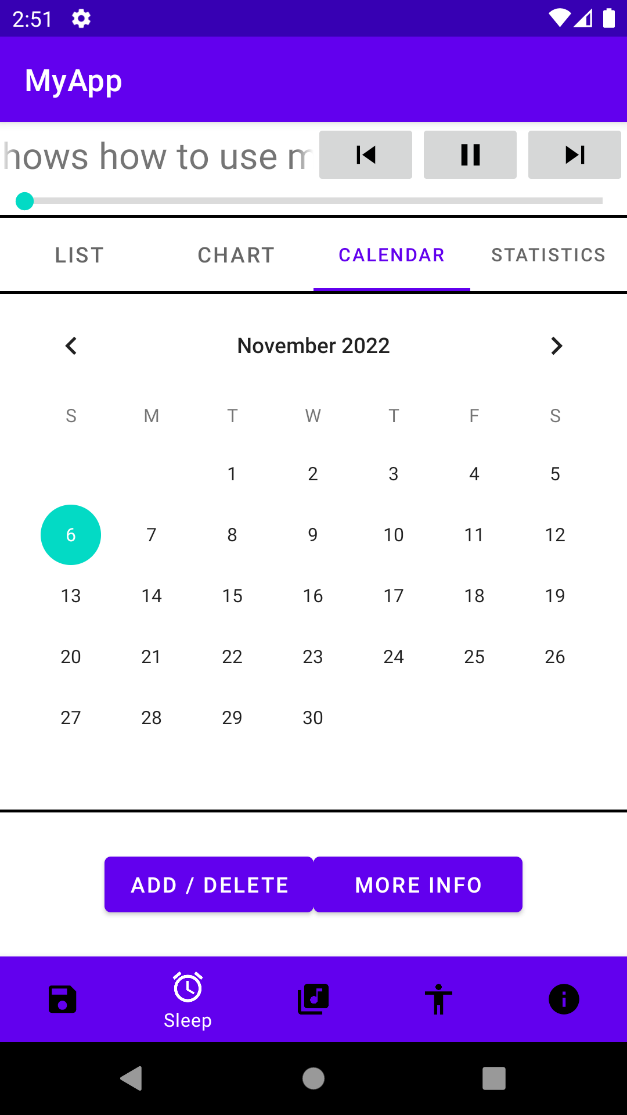
This is the second **fragment** of sleep activity. [**Linear**]

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**. 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 third **fragment** of sleep activity. [**Linear**]

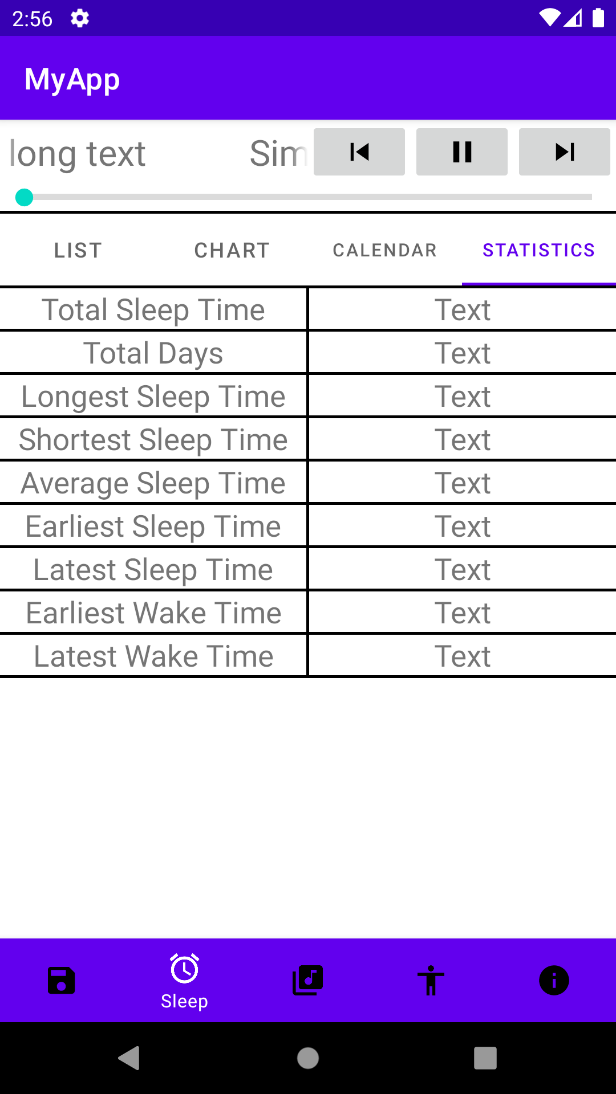
The **broadcast receiver** notifies the fragment if there is a change in sleep list data.

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**.



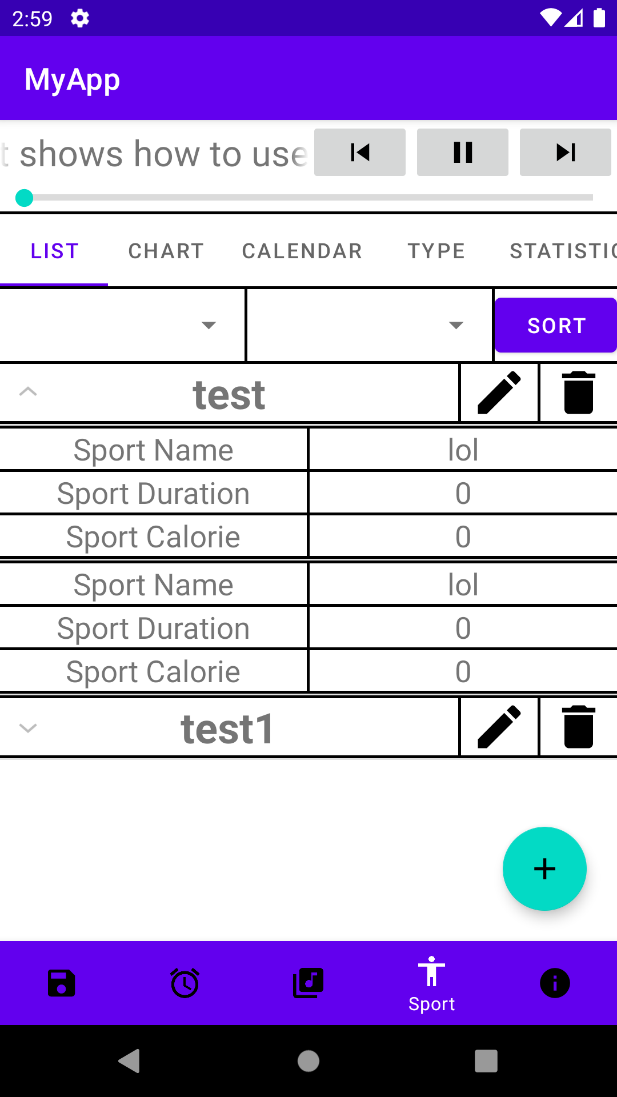
This is the fourth **fragment** of sleep activity. [**Linear**]

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 first **fragment** of sleep activity. [**Relative**]

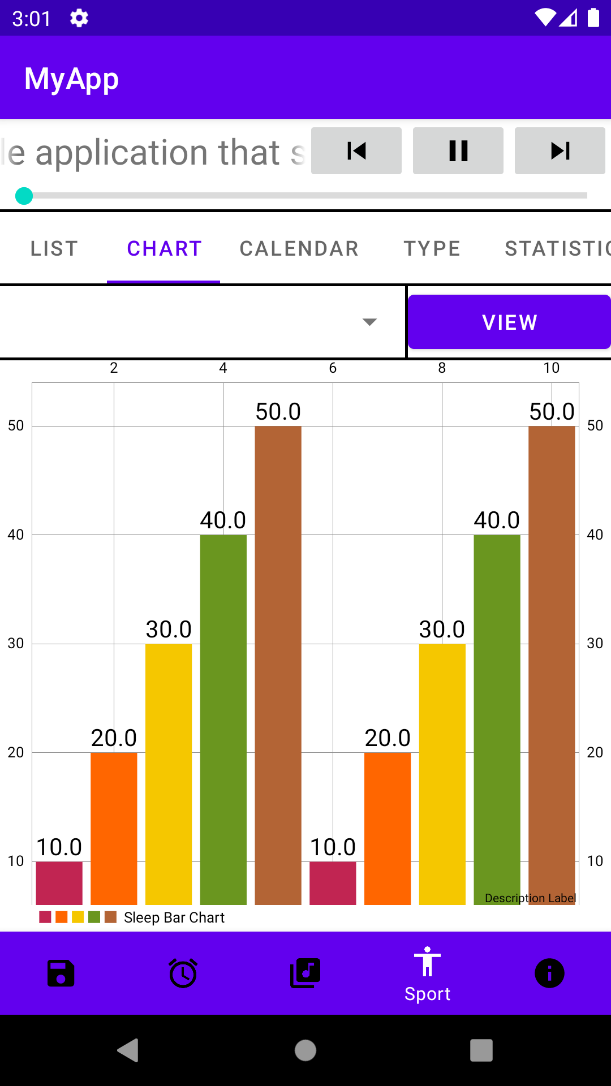
The **broadcast receiver** notifies the fragment if there is a change in sport list data.

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 button** is used to redirect users to another activity to create new sport data.

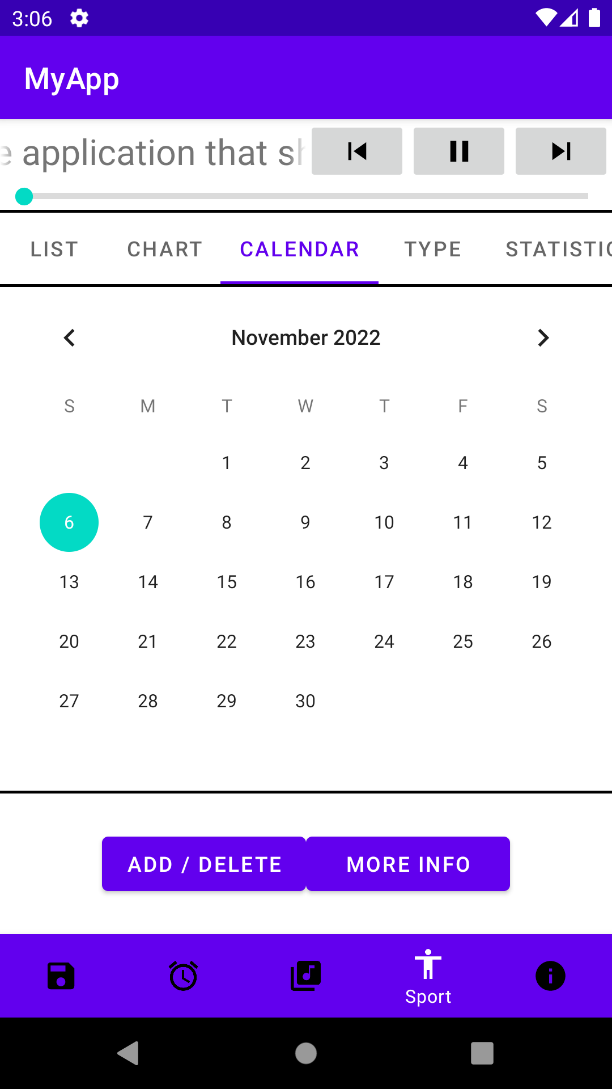
This is the second **fragment** of sport activity. [**Linear**]

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**. 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 third **fragment** of sport activity. [**Linear**]

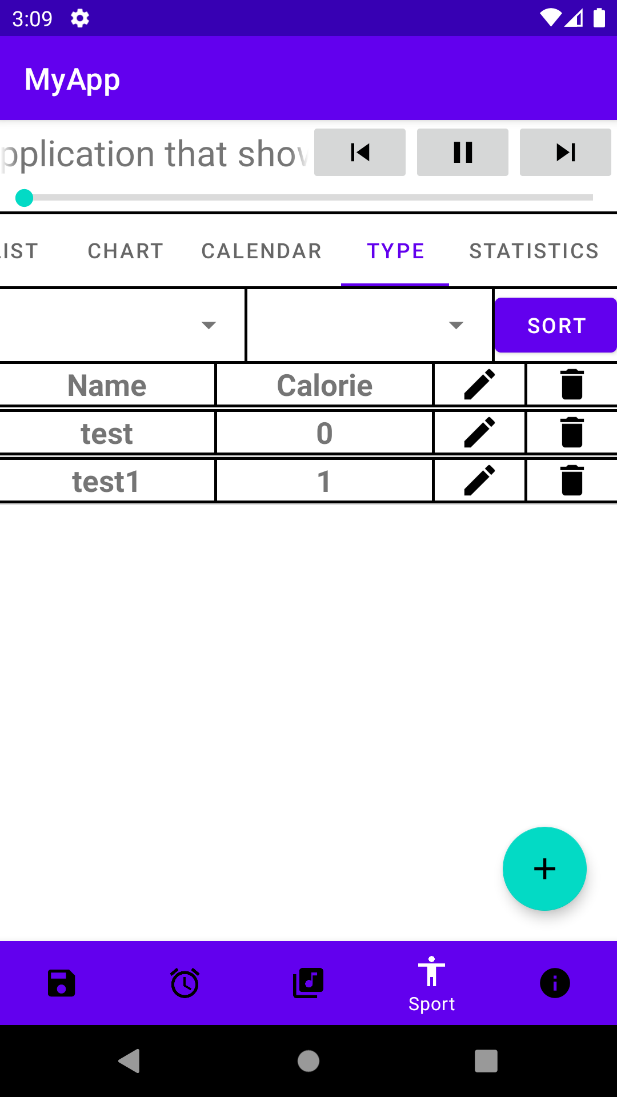
The **broadcast receiver** notifies the fragment if there is a change in sport list data.

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 fourth **fragment** of sport activity. [**Relative**]

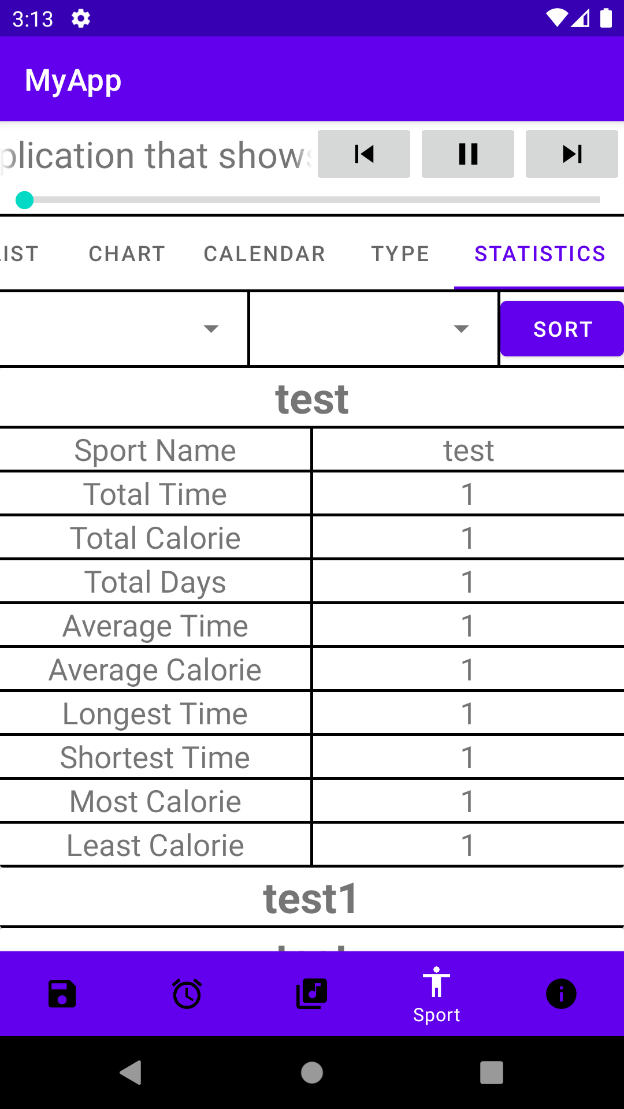
The **broadcast receiver** notifies the fragment if there is a change in sport type list.

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 button** used to add new sport types to the app by calling a **custom dialog**.

This is the fifth **fragment** of sport activity. [**Linear**]

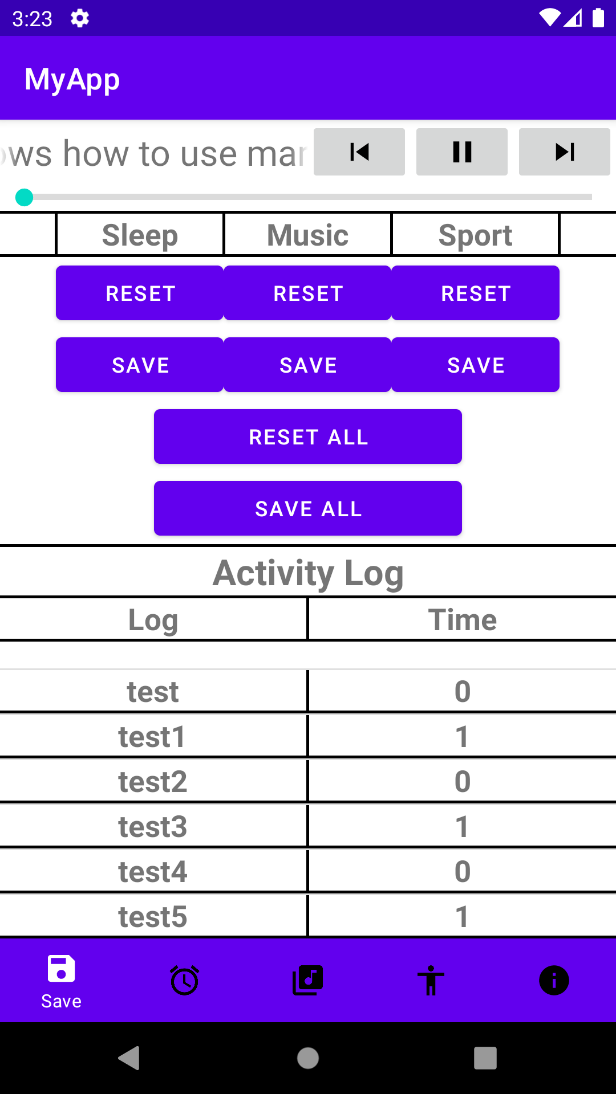
The **broadcast receiver** notifies the fragment if there is a change in sport list data.

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**]

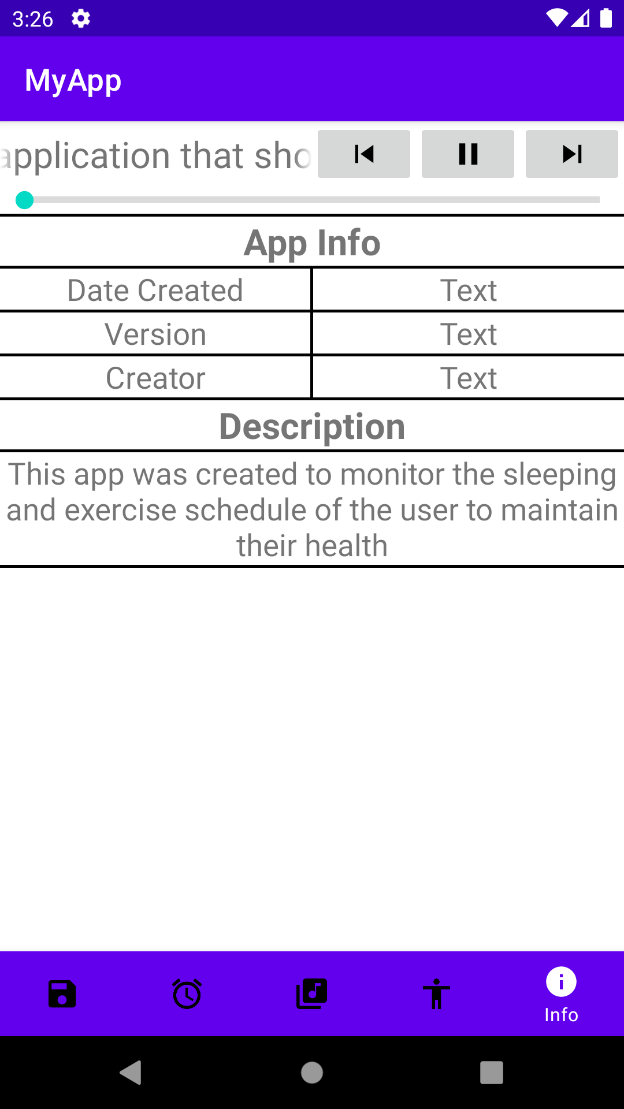
It has a mp3player on top with **image buttons** and a **seek bar**. It also has a **bottom navigator** to navigate to other activities.

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.

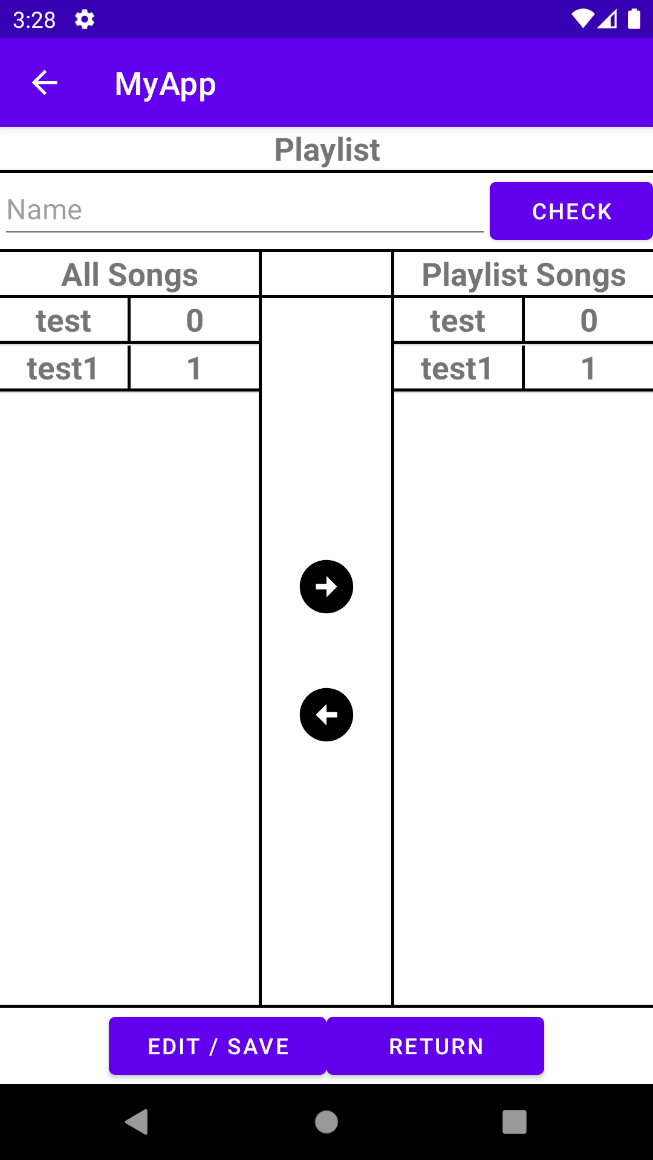
The user can reset any changes made to the data using the reset **button**. Users can also overwrite the old data with new data with the save **button**. The reset all and save all **button** are also provided for ease of use. This means that saving is manual in this app and users must remember to save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

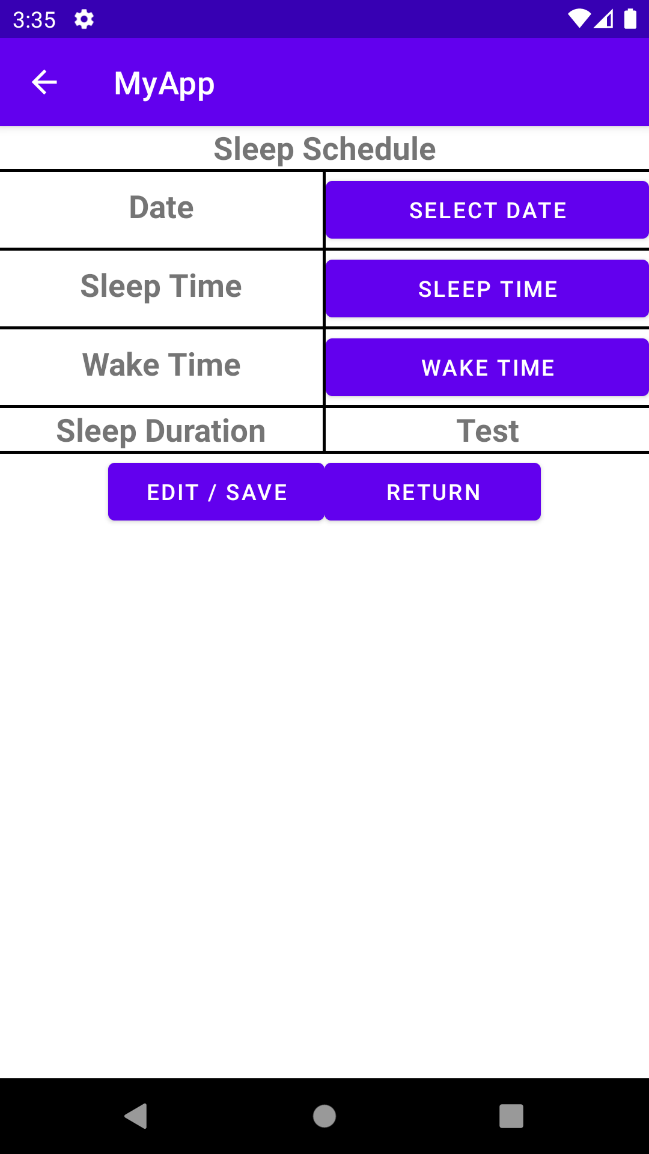
The **broadcast receiver** notifies the activity if there is a change in song list data.

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 Creation **activity**. [**Linear**]

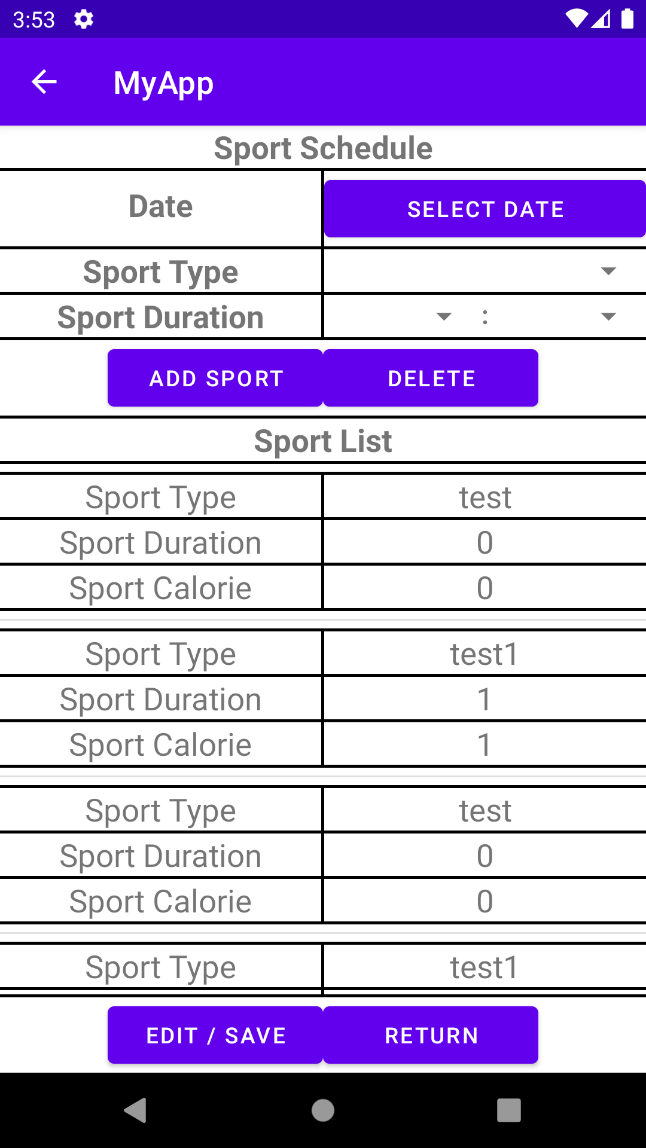
The **broadcast receiver** notifies the activity if there is a change in sleep list data.

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 Creation **activity**. [**Linear**]

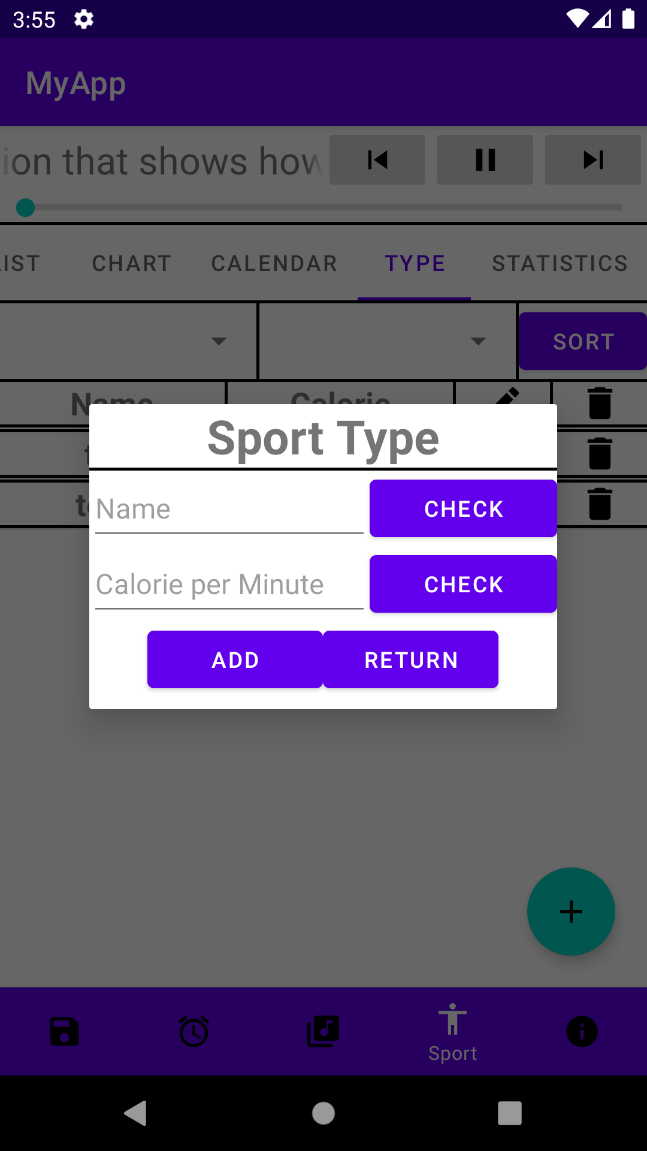
The **broadcast receiver** notifies the activity if there is a change in sport type data.

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 sport 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 Creation **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**.