

**CSCU9YH – Unit Conversion App**

Student ID: 2520796

# **Introduction**

The purpose of this project is the development of an Android Unit Conversion application using Kotlin language. The application should convert units e.g. distances, weights, speeds etc. and should be able to accept a user input for the given unit (e.g. Distance–Meters) and convert it to the target unit (e.g. Distance-Kilometers).

The approach used concerns the creation of both fragments and a User Interface that is user friendly and extremely easy to use even without the aid of any instructions. In this respect, it should include some features to be as user friendly and attractive as possible to the user, for example, a Day-Night theme, a Spinner menu that shows the units to select or a navigation menu that appears from left to right when slide. Moreover, it may include a Settings button to let the user have an option to tweak some settings, such as, choose:

* how many digits should appear to the conversion result or increase/decrease the general font.
* orientation compatibility.
* automatic refresh screen with the results instead of pressing the conversion button every time.
* more conversion units (e.g. Force, Radiation, Energy and online Currency/Cryptocurrency conversion etc.).
* icons for each unit and/or background colors.

Figures 1 and 2 illustrate the very first version of the mobile application.

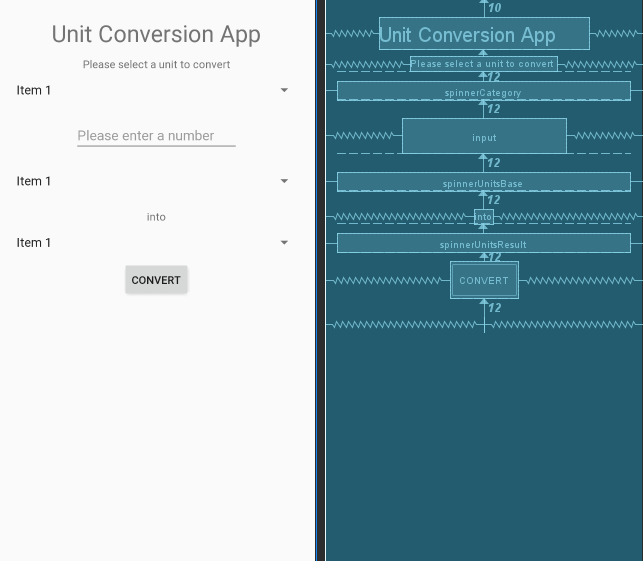


Figure 1: Version 1 Design

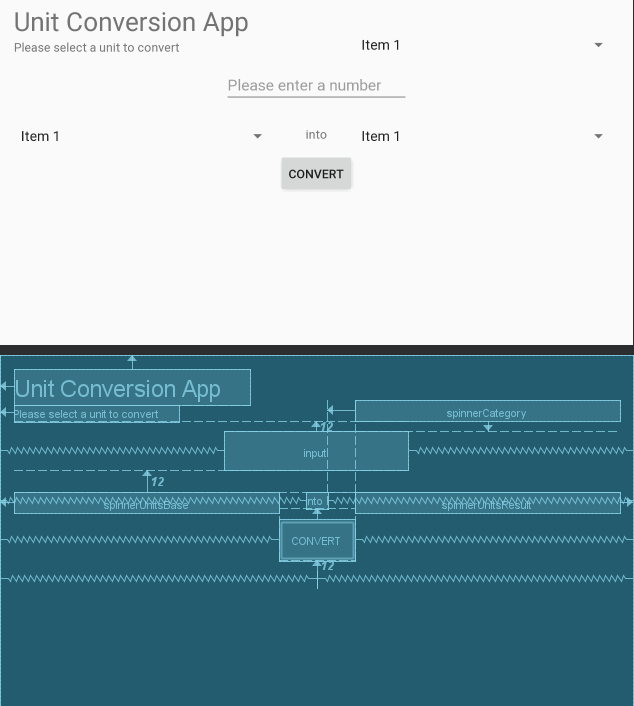


Figure 2: Version 1 Design Landscape

# **Structure of the app**

The first version of a Unit Converter app includes two fragments, and three spinners, one of which is the Quantity of the unit (e.g. Length, Volume)and the other two are the units that are responsible for converting when the Convert button is pressed (e.g. Meters into Kilometers). Moreover, it includes one button that converts the selected units and three text fields that display the Units. The construction of the application was very simple this way, therefore, the development of a second improved version was needed (and this one was discarded)as a proper menu with icons next to each Unit, background colors, grids and more organized text fields could be way more enjoyable and effective for the user experience. Poorly and long menus could be difficult or complicated for the users, whilst, a plain white background with couple of simple text fields with the aim of just getting the job done could be complex and not pleasant for the user experience.

# **UI Design and features**

The Android app’s UI is a hierarchy of templates and widgets. The designs and artifacts of the View Group are simply containers that monitor the location of their child views on the display. Widgets are UI elements include buttons, text fields and generally are objects for display.

## **First State**

The app possesses important features that will improve the user experience. To begin with, the first thing the user sees when the open starts is a big empty editable text field that prompts the user to input something, while the below smaller text fields will provide the results of the other units to the user all at the same time automatically without the use of a “Convert” button. The figure below presents the first state of the application.

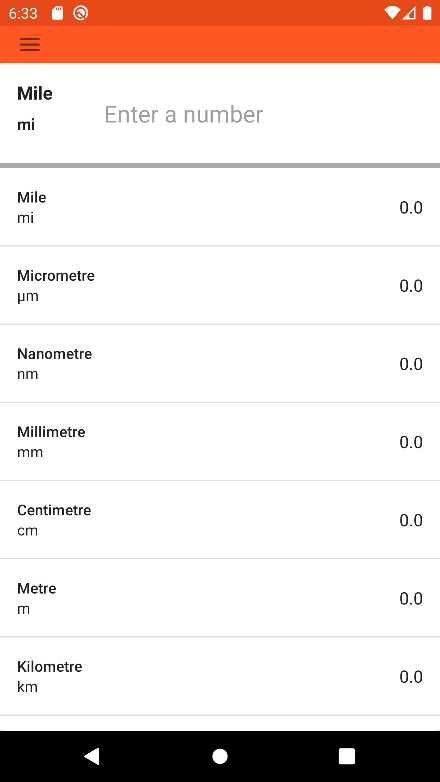
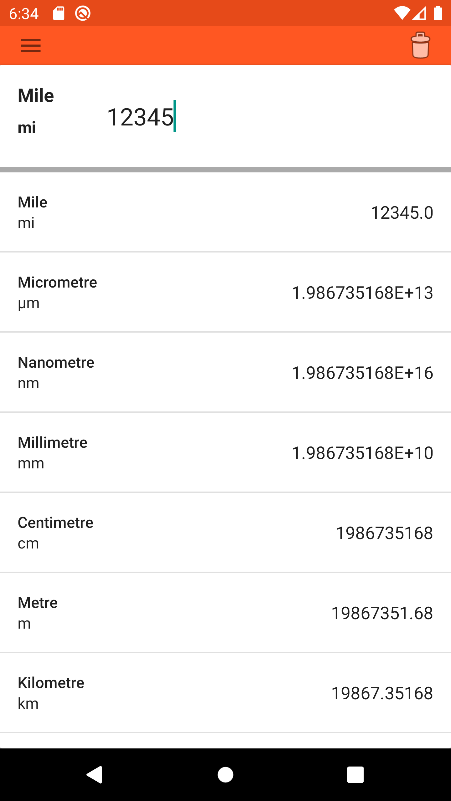
 

Figure 3: First state of the application Figure 4: Real Time conversion

The first state includes, among others, the selected Unit with its symbol, which in this case is Mile, and, on the center, there is an empty editable text box “Enter a number” which is the place that the user inputs a decimal negative or positive value. The conversion occurs in real time and the results produced will replace the 0.0 values of each of the below units correspondingly (Figure 4). The user can tap any of the below Units (e.g. Centimeter, Meter etc.) to select a Unit or can long tap any of the Units to copy that unit’s value to clipboard. Furthermore, the “trash can icon” to the upper right corner is the clean button that cleans the input value.

## **Sliding Menu**

In comparison with the previous version of the application, a side sliding menu replaced the spinners. The user can slide from the left side of the screen to the right or just press the upper right hamburger shaped icon to open the menu. The menu contains all the unit categories available for conversion (e.g. Area, Data Transfer Rate, Mass etc.) as shown in the Figure 5.

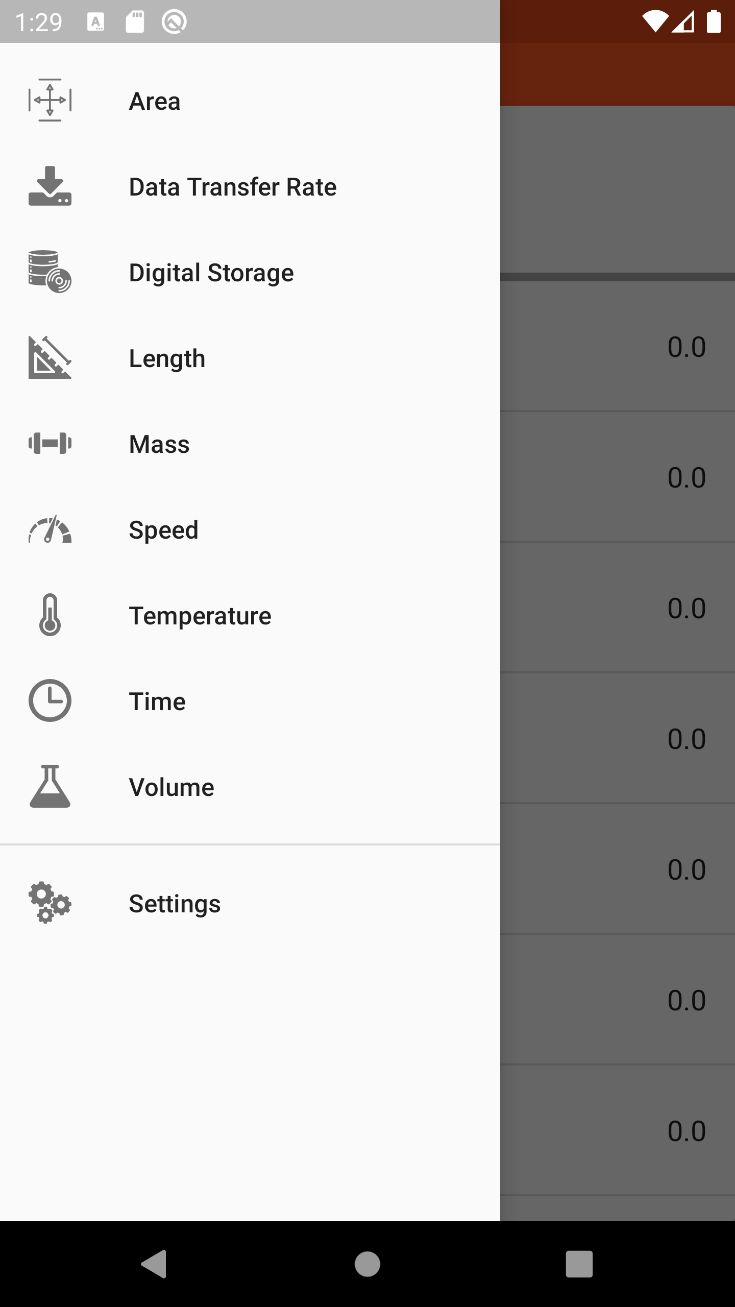


Figure 5: Side Menu

## **Orientation**

Orientation has been applied to the underlying Android application to test the functionality. The test has been applied to most of android devices (tablets, smartphones, older generation of smartphones) using a variety of Android API on Android Virtual Device and has been tested on real android devices.

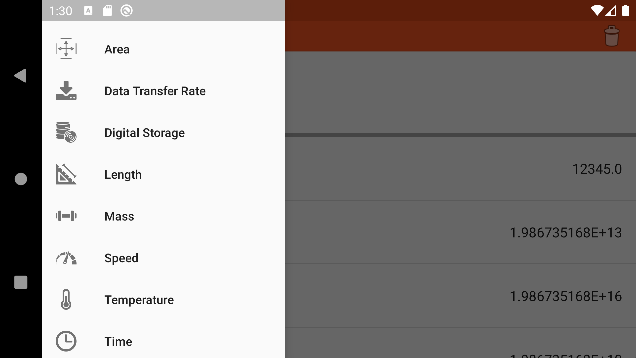
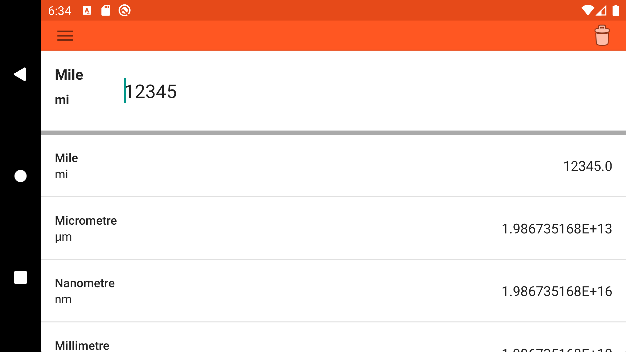


Figure 6: Landscape Main & Menu

## **Settings Menu**

The settings menu contains only one extra feature now which is giving the option to the user to change the theme of the app. The user can change the theme into Normal Mode or Dark mode as shown to the Figure 7. Moreover, a “Back Arrow” button is located to the upper left side of the display which takes the user back where it was.

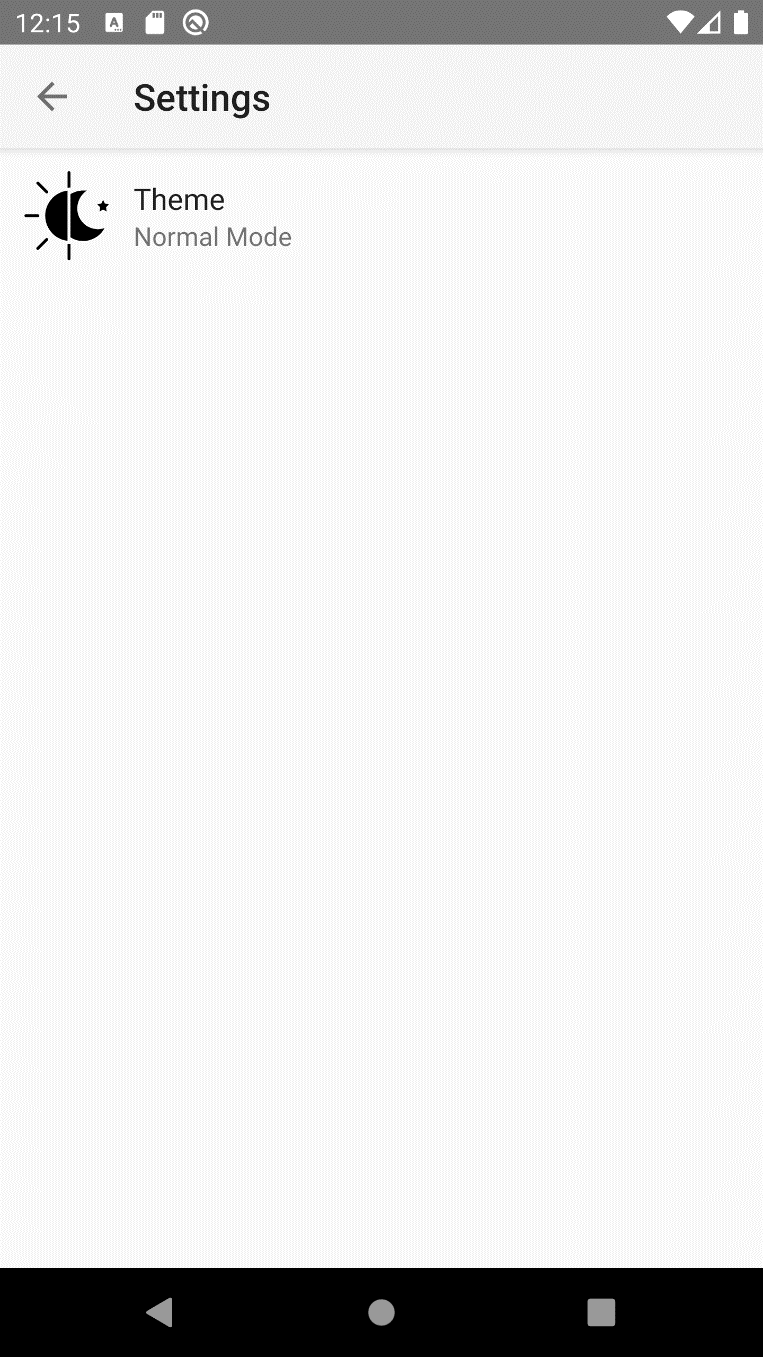


Figure 7: Settings Menu

## **Dark Mode**

When the user selects to switch to Dark mode the app will pop up a message to restart the app in order to activate the dark mode. The dark mode theme is not different in any functional way from Normal Mode except the background, icon and text colors which have been switched to the opposite of the Normal mode (e.g. White background -> Black background, Black text -> White text etc.). Figures 8 – 11 demonstrate the application in Dark Mode.

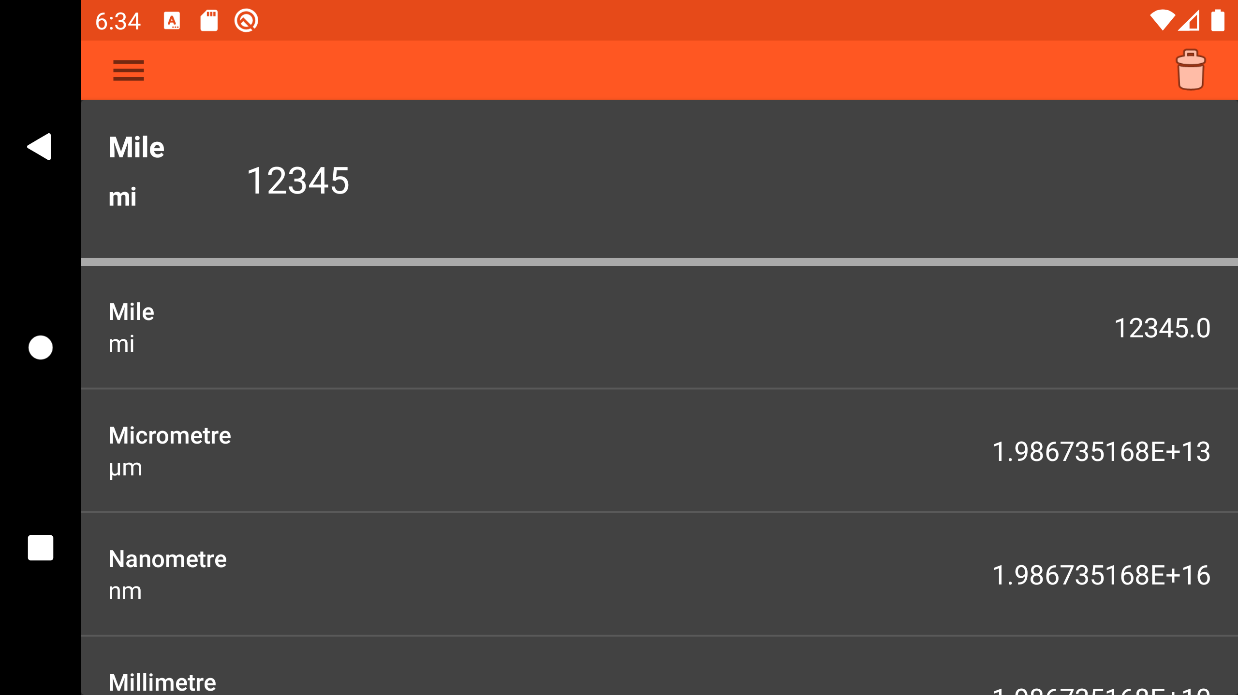


Figure 8: Dark Mode Landscape Main

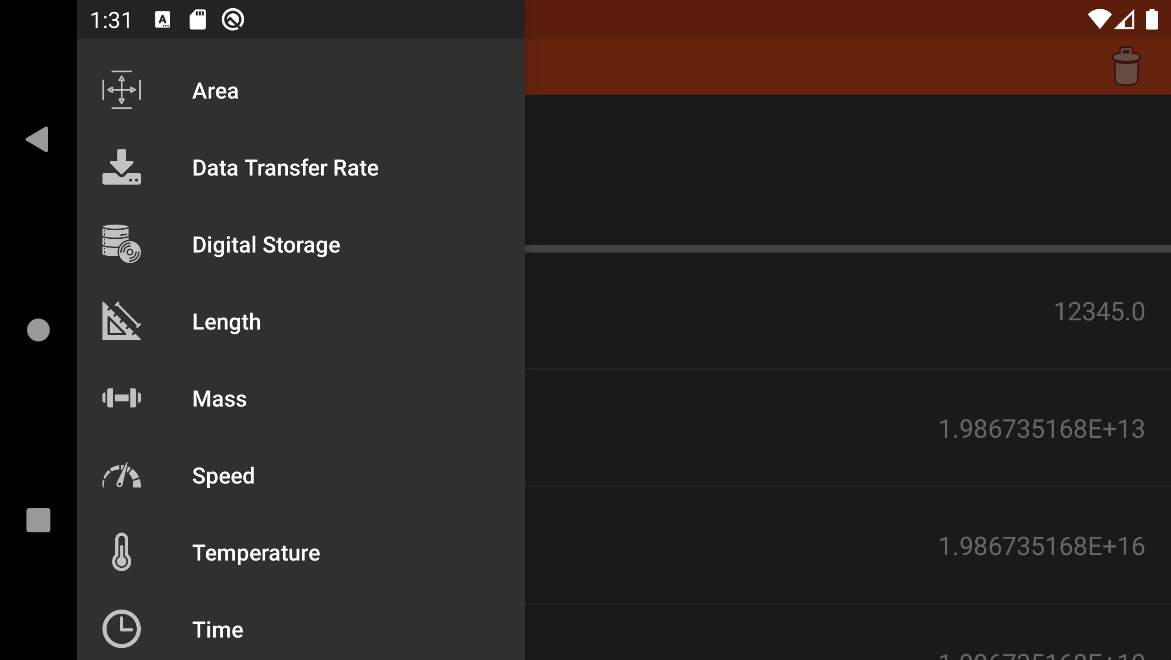


Figure 9: Dark Mode Landscape Menu

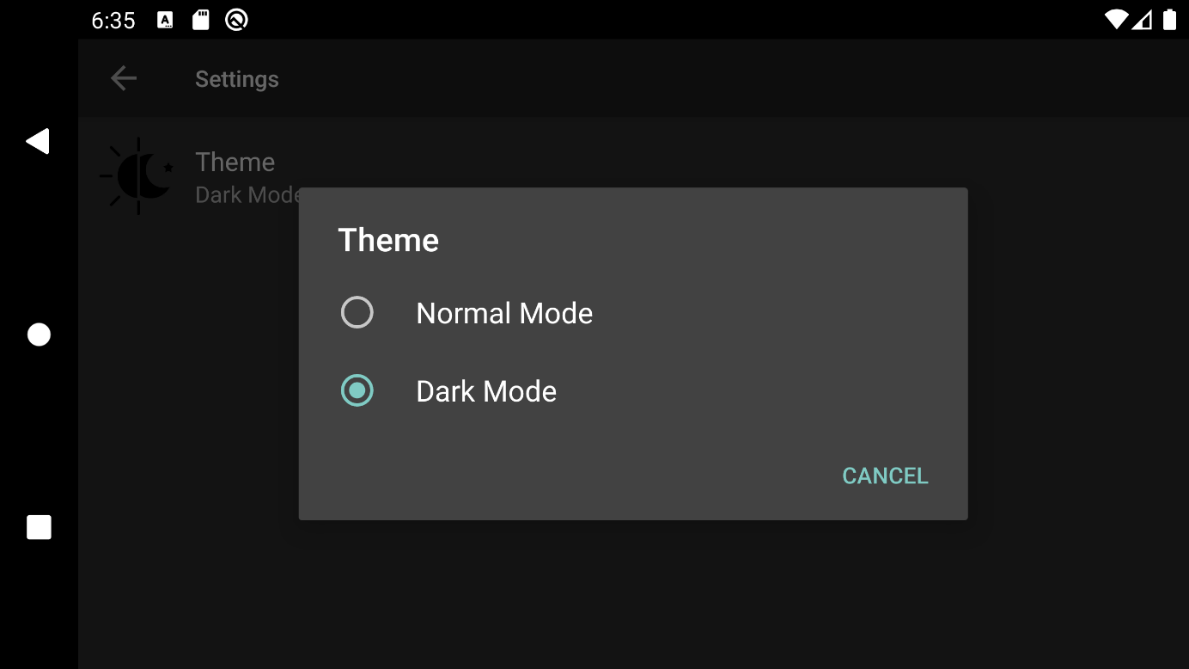


Figure 10: Dark Mode Landscape Theme Selection

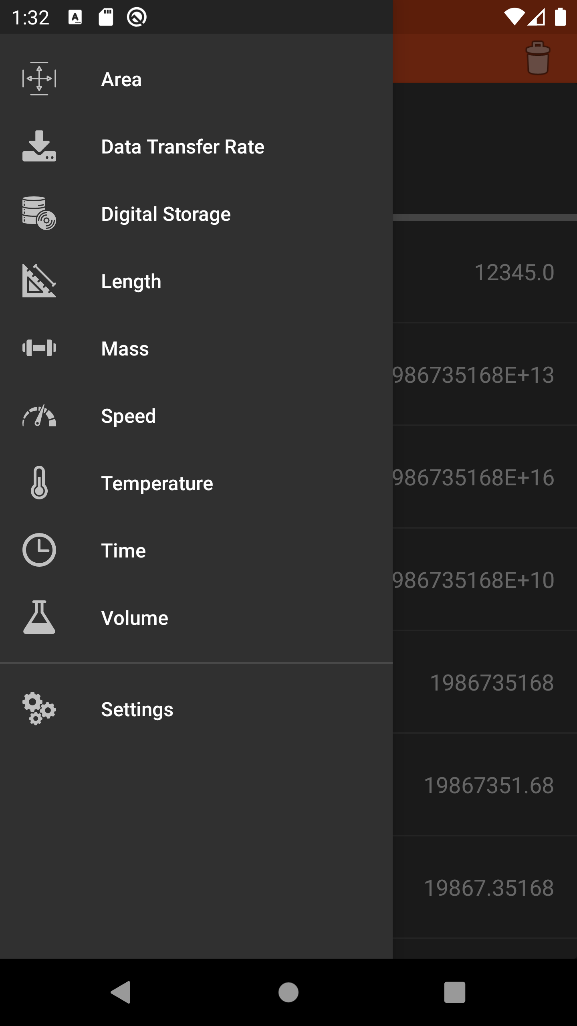
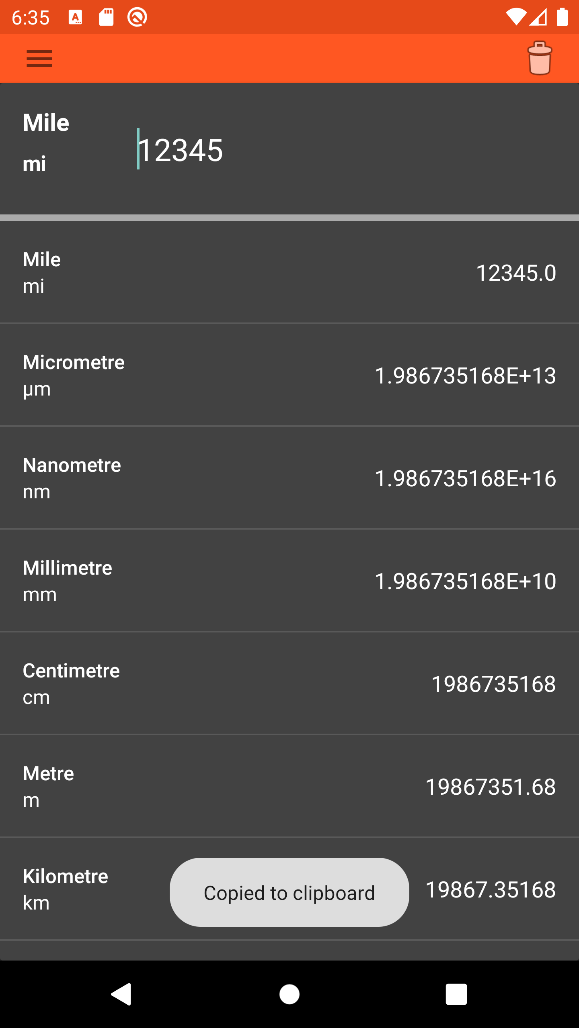


Figure 11: Dark Mode Main and Menu

# **Code Listings**

This application was developed in Kotlin language instead of Java for experimental purposes. Below is a list with all the resources xml files:

1. **Drawable:**

|  |  |
| --- | --- |
| Icon\_area.xml | Vector icon for the Area unit |
| Icon\_back\_arrow\_button.xml | Vector icon for the back-arrow button in the settings menu section. |
| Icon\_data\_transfer\_rate.xml | Vector icon for the Data Transfer Rate unit |
| Icon\_digital\_storage.xml | Vector icon for the Digital Storage unit |
| Icon\_length.xml | Vector icon for the Length unit |
| Icon\_mass.xml | Vector icon for the Mass unit |
| Icon\_menu\_hamburger.xml | Vector icon for the Menu button |
| Icon\_night\_day\_theme.xml | Vector icon for the Dark Mode theme in the settings section |
| Icon\_settings.xml | Vector icon for the settings |
| Icon\_speed.xml | Vector icon for the Speed unit |
| Icon\_temperature.xml | Vector icon for the Temperature unit |
| Icon\_time.xml | Vector icon for the Time unit |
| Icon\_volume.xml | Vector icon for the Volume unit |

1. **Layout:**

|  |  |
| --- | --- |
| Sample\_layout.xml | This is an empty Sample Layout design which it can be used in a more useful way with Anko-design. |

1. **Menu**

|  |  |
| --- | --- |
| Clear\_button.xml | This xml handles the Clear Button |
| Side\_unit\_menu.xml | This xml has the basic design of the side menu. It contains 2 groups. One the Units Menu which is contains all the items (units) and the other group is the Other Menu which contains the Settings menu. |

1. **Mipmap**

|  |  |
| --- | --- |
| Ic\_launcher.png | This image is the application icon that is going to appear on the user’s phone after the application is installed. |



Figure 12: Launcher Icon

1. **Values**

|  |  |
| --- | --- |
| Colors.xml | This xml contains a small list of specific colors |
| Strings.xml | This xml contains all the strings of the application |
| UI.xml | This xml contains some settings for the default UI (e.g. bar\_height, margin etc.) |

1. **Xml**

|  |  |
| --- | --- |
| Night\_day\_theme.xml | This xml contains the settings for the Dark Mode feature |

Below is a list with all the Kotlin files of the application:

|  |  |
| --- | --- |
| Activity.kt | * Displays the SettingsUI Layout created through XML. * Save instance so it can restore to previous state. * Manages a fragment |
| Conversions.kt (Object) | * MutableMap holds the Units in a form of a key and a value pair (name, symbol, value) * Send the list to the UnitConversion for calculation |
| Fragment.kt | * Save instance * Add a preference from the Theme and prompts the restart |
| Recycler.kt | * Reference items from layout file |
| RecyclerContext.kt | * Handles the single tap (selection of unit) and long tap (copy the value) * Handles the user input and result. Exports results toBigDecimal |
| SettingsUI.kt | * Creates the UI for Settings (back arrow icon, layout, frame etc.) |
| UnitCalc.kt (Interface) | * This is an abstract method for the UnitConversion.kt |
| UnitConversion.kt | * Handles the conversion with an equation. Takes the input, it converts it to base and then back. * F(x) = [a] \* x^[n] + [b] |
| Units.kt (Enum) | * Enum class that contains a list with R.id of all units. R is public final class which extends Object class. |
| UnitsUI.kt | * Creates the UI of units using anko through the sample\_layout.xml |
| MainActivity.kt | * When the application start onCreate is initialize the activity and sets the Theme, AndroidToolbar, MiniFeatures, UnitMenu, and savedInstances. * onResume() method can be called through the activity lifecycle and is counterpart to onPause() * It handles the input of the user to be saved when an orientation happens, when the user restarts the app through the Theme switch and when another category is selected. * It sets the colors of the toolbar and layout * It handles the copyToClipboard text |

# **Additional Functionality**

Finally, up to this stage, the application is working smoothly without any crashes or bugs. However, if there was more time, additional features could be implemented to improve the current app even further.

## **More Categories (Static and Online)**

Currently, there are nine categories for the user to choose from and some more could be added. For example, a Pressure category with Atmosphere, Bar, Pascal or Torr units to convert or an Energy category with Joule, Kilojoule, Watt Hour, Electrovolt etc. The more categories and units included in the application the more complete it will be. Moreover, due to time limitation two of the most important categories were not included in this project: Currency and Cryptocurrency. The reason behind their exclusiveness relies on accuracy, as it would be inaccurate if the conversion rate is a static/fixed number. An online connection will be required for the program in order to ensure access to an online database and get in real time the exchange rate for each currency.

## **More Settings**

As mentioned above, there is only one feature in the Settings section, the Dark Mode theme. A good upgrade for the application is to include more options, such as:

* An option to change the Font Size.
* An option to change the digits after the decimal point.
* An option to include an on-screen keypad instead of the android stock one.
* An option to hide/unhide units. For example, in the Length category a user may need to only see the Meters, Kilometers and Centimeters instead of every single unit at the same time.
* An option to change the Language.
* An option for the user to send a feedback or report a bug. If the application is published this is a good way to get reports and feedbacks from users, thus, the developer could be informed of other opinions or get ideas and fix the bugs for the next update.

## **Make the application Hybrid**

The current app is made specifically for android devices. Even though the current app is a native app, in the future, it can become a Hybrid one by using Cordova or Ionic technologies to release the app to more platforms e.g. iOS, Windows.

## **Monetize the application (ads or premium features within trial time)**

Monetizing an application could be very handful and profitable for the developer. For example, some small and not annoying banners with ads relevant to the topic of the app or 1-month trial with some premium features is a good strategy to let the user experience the app first before making the purchase.

## **Publish the app to Google Play Store**

Finally, as a developer, is a very good start to publish an application to Play Store, which, provides a lot of amazing benefits. Furthermore, most of the android users use Play Store to download their favorite apps, thus, if it becomes successful it may become popular and the download number will keep increasing.