CSCU9YH – CONVERSION APPLICATION REPORT

Student ID Number – 2636157
University of Stirling

Contents

1. Introduction

For my android assignment I was given the task of developing a unit conversion mobile application using the programming language Kotlin in the Android Studio. The applications functionality requirements were to allow the user to select two different values (e.g. GBP to USD) they want to convert from and to from a selected unit (e.g. Currency).

Once the user has selected two conversion values, they will now have the ability to move to a different section of the application which will shows the user what values they have selected and allow them to input the a numeric value that they want to convert from. This is done by an inbuilt calculator that provides the numeric input along with the ability to clear the input from the calculator.

The application also requires implementation of at least two different fragments within an activity, this being a fragment for selecting the unit, value, and conversion and the second being the conversion calculator. The layout of the application must be very straightforward and easy to understand without the use of help and tips to work the application. Also, there must be at least four different sets of unit conversions (e.g. GBP to USD, EUR to JPY, etc…).

2. Application Structure

The application that I have developed consists of three different fragments, one for each of the different actions the user can carry out. Each fragment contains a constraint layout along with tables to hold all the information and features being displayed. This provides a clean and simple look, ensuring that everything is being displayed correctly. The application as includes a database that handles the distribution of the conversion data throughout the application.

Fragment One

Fragment one is the first fragment that is displayed when the application is launched, on display are three different spinners, the first one being the unit spinner. This spinner allows the user to select a unit (e.g. Currency) that will then allow them to use the other two spinners, the value (convert from) spinner and the conversion (convert to) spinner. These two spinners display the values of the unit selected (e.g. GBP, USD, EUR) that get passed into the second fragment.

Screenshots of fragment one xml design is shown in section 2.1. Design Screenshots *Figure 2* and *Figure 3*.

Fragment Two

The second fragment is displayed when the user navigates over to the second fragment, once the fragment has been switched, the two values that the user selected are displayed under value and conversion selected. The user then must use the inbuilt calculator to enter a numeric value, after they have entered a value and pressed the convert, the conversion is done automatically and gets displayed back to the user.

Screenshots of fragment two xml design is shown in section 2.1. Design Screenshots *Figure 4* and *Figure 5*.

Fragment Three

The final and third fragment included in the application holds the functionality of the database. The user has the option to insert a new conversion into the database which they can then use to carry out their own conversions. Creating your own conversion consists a number requirements that the user must input, these are the unit, value (convert from), multiplier and the conversion (convert to). When a new insert has been made, all text field get cleared and ready for the next conversion to be created. The contents of the database also get displayed to show that the new conversion has been inserted into the database with the use of a scroll view. If the database content is not already displayed or the user just wants to look at the contents without inserting a new conversion, the user can press the display button instead.

Screenshots of fragment three xml design is shown in section 2.1. Design Screenshots *Figure 6* and *Figure 7*.

2.1. Design Screenshots

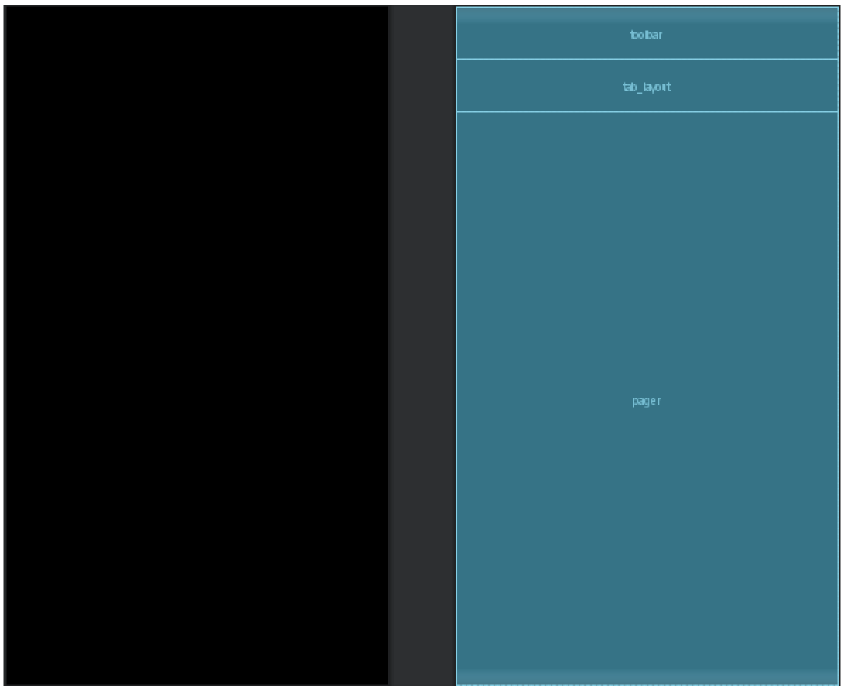


Figure 1.
Activity Main XML File

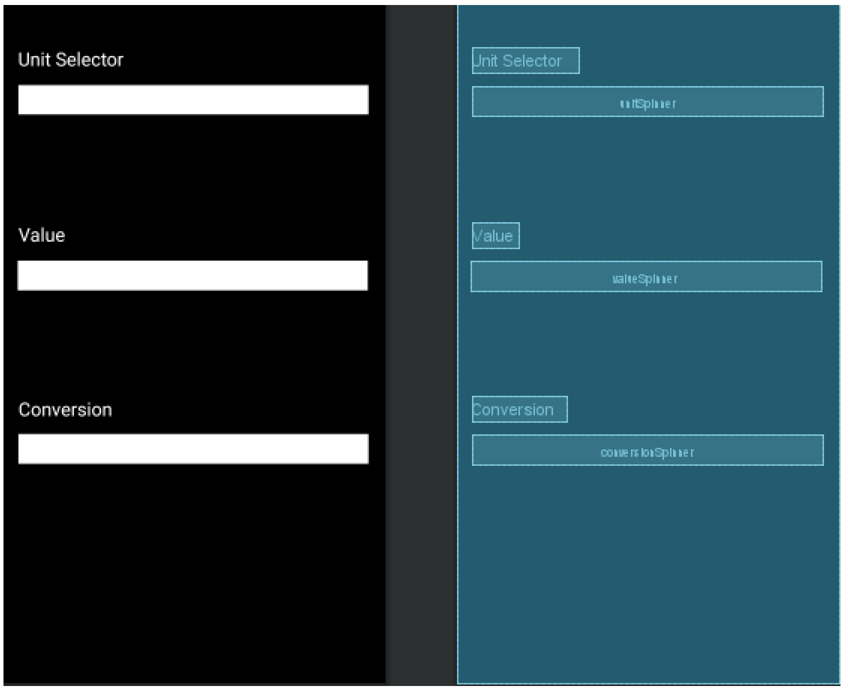


Figure 2.
Page One Fragment XML File

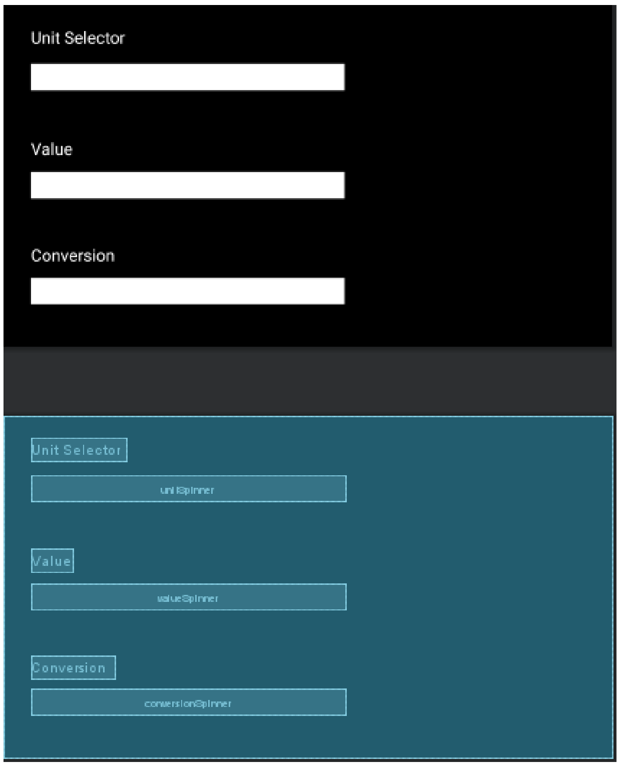


Figure 3.
Page One Fragment Landscape XML File

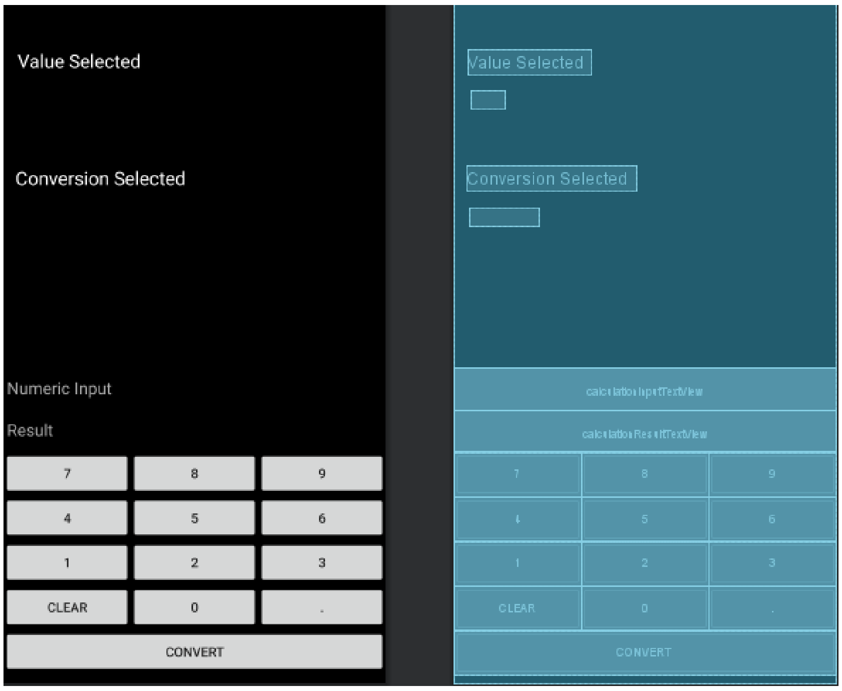


Figure 4.
Page Two Fragment XML File

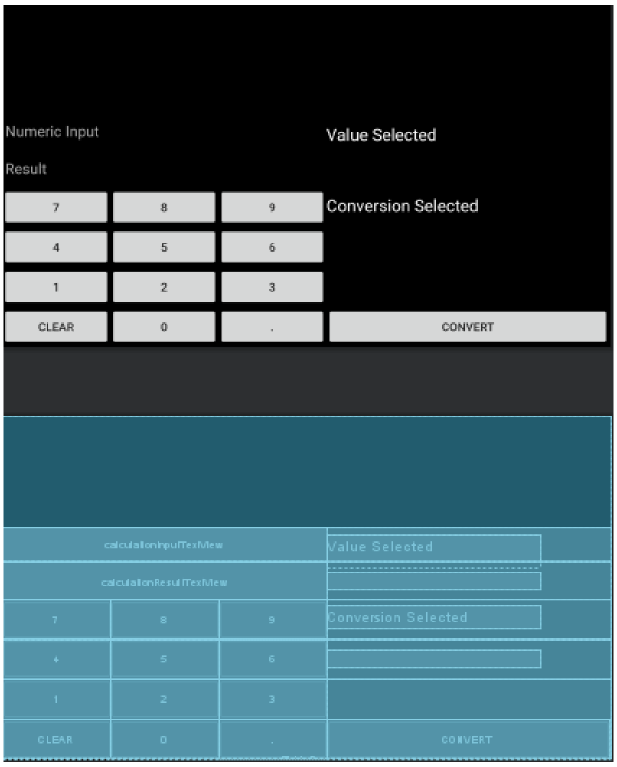


Figure 5.
Page Two Fragment Landscape XML File

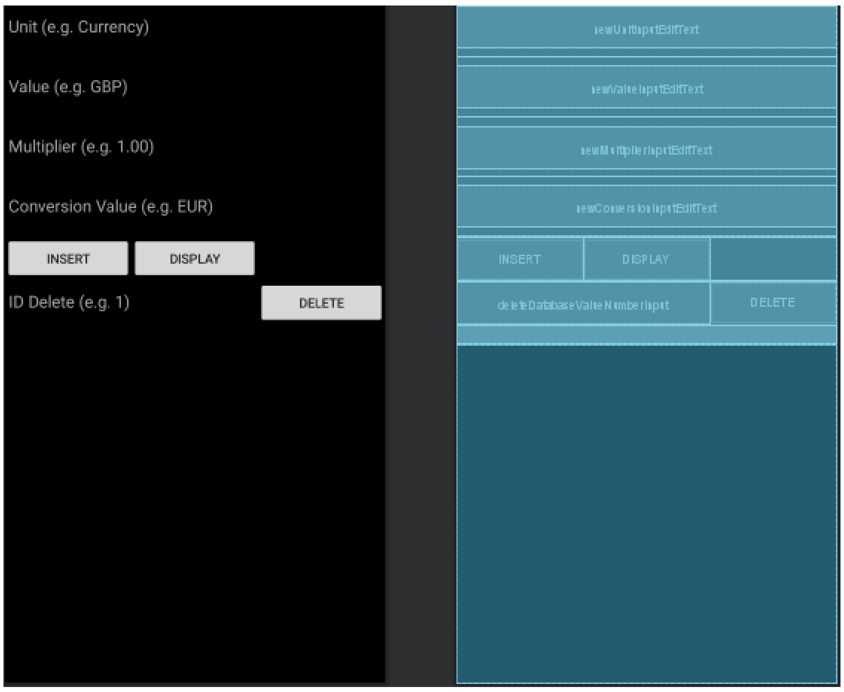


Figure 6.
Page Three Fragment XML File

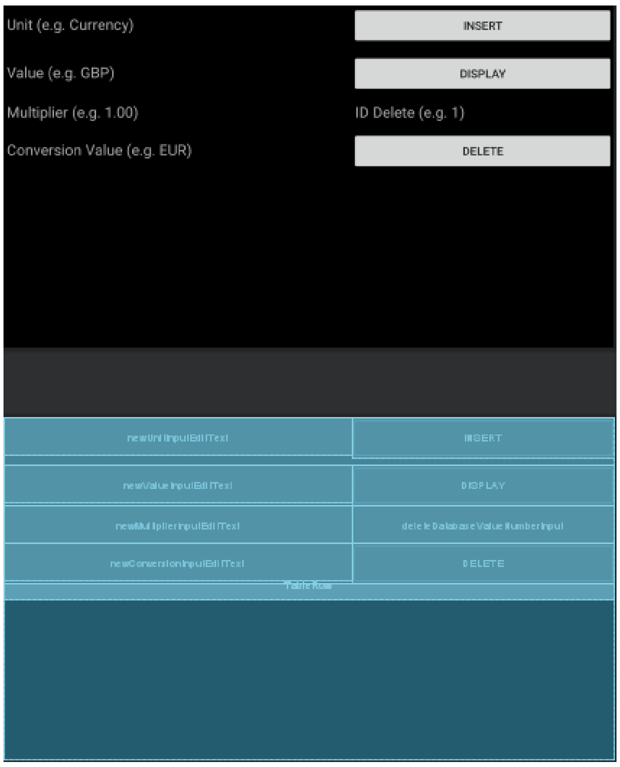


Figure 7.
Page Three Fragment Landscape XML File

3. UI Design

3.1. Screen Orientation

I was able to implement screen orientation into the application, the application can function in both portrait and landscape modes. With the implementation of the database to distribute the data to the spinners, the current selection ion each spinner stays the same when the screen is rotated. I tested the application in an emulator and on a physical mobile device and both versions worked correctly with no errors or bugs.

Screenshots of each fragment being in landscape mode is shown in section 3.6. Application Screenshots *Figure 9*, *Figure 11,* and *Figure 13*.

3.2. Colour Scheme

The colour scheme I have went with was a dark theme, I chose this because it makes easy to read the text as the colours of black and white contrast well with each other. This done for everything such as the text, buttons, and the tab selector to make sure that the colour scheme throughout the application is consistent to provide a professional looking layout.

Screenshots for all the fragments are show in section 3.6. Application Screenshots.

3.3. Navigation

The process that the user carries out to navigate through the application is by using tabs located at the top of the screen. This allows the user to move back and forward throughout each of the three different fragments. The tab the user is currently on being displayed is highlighted in the tab menu with an underline. Using fragments with the tab switcher provides a quick and easy transition process throughout the application.

A screenshot of the tab is shown in section 3.6. Application Screenshots *Figure 19*.

3.4. Error Messages

The way that the application deals with errors is with toast messages, these messages pop up at the bottom of the screen when an error had occurred. This method is very quick and efficient way of notifying the user that something has gone wrong. It also gives the ability of display any message you wish with a specific message to alert the user about what the error is and what can be done to resolve it.

Screenshots of some error messages are shown in section 3.6. Application Screenshots *Figure 14* and *Figure 15*.

3.5. Functionality on a Tablet

I decided to test the application on a larger device such as a tablet to see how the designed layout would handle a change in screen size. This was done through an emulator of a tablet and produced expected results of handling the change well even through some of the features were slightly out of place, it still resembled the design. There was no layout created for tablet devices, so the design generated was based of the already created xml files.

Screenshots of the tablet displays are shown in section 3.6. Application Screenshots *Figure 16*, *Figure 17,* and *Figure 18*.

3.6. Application Screenshots

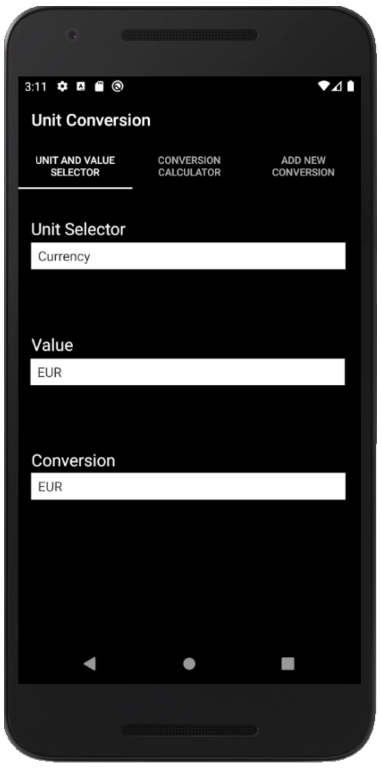


Figure 8.
Page One Fragment from the Emulator

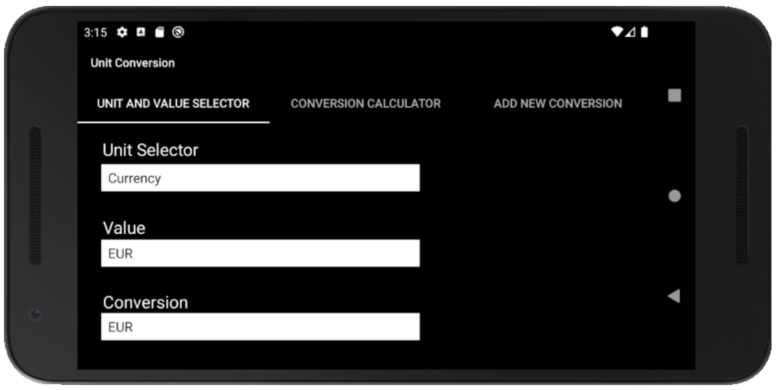


Figure 9.
Page One Fragment Landscape from the Emulator

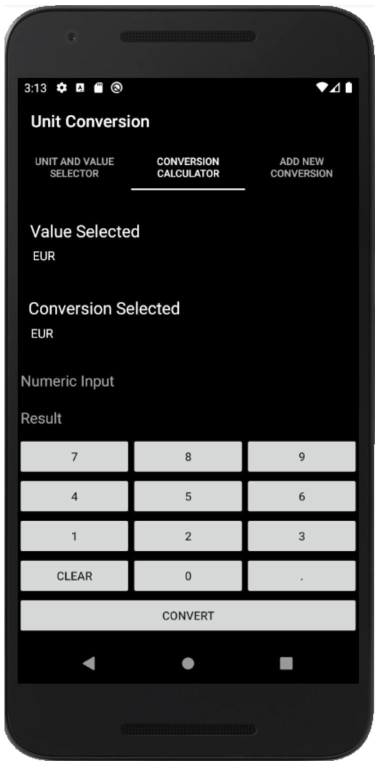


Figure 10.
Page Two Fragment from the Emulator

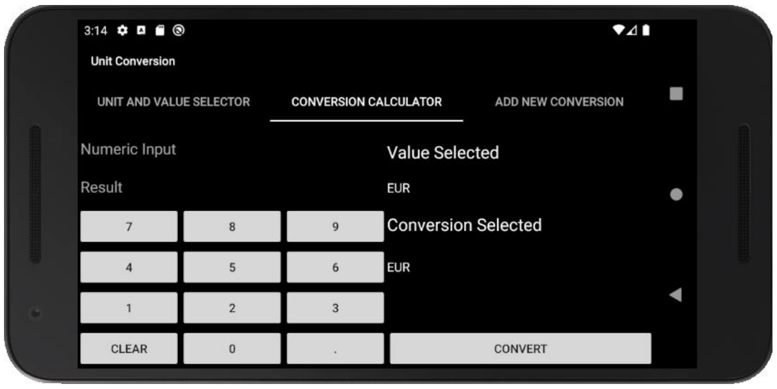


Figure 11.
Page Two Fragment Landscape from the Emulator

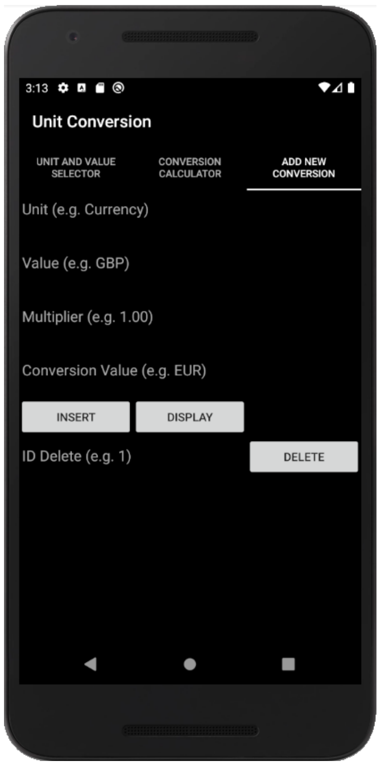


Figure 12.
Page Three Fragment from the Emulator

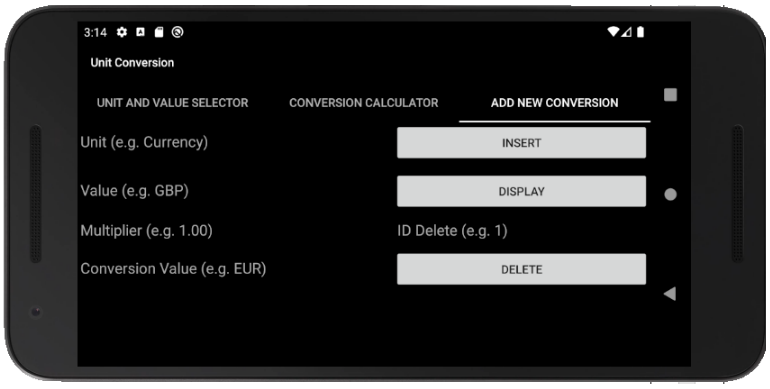
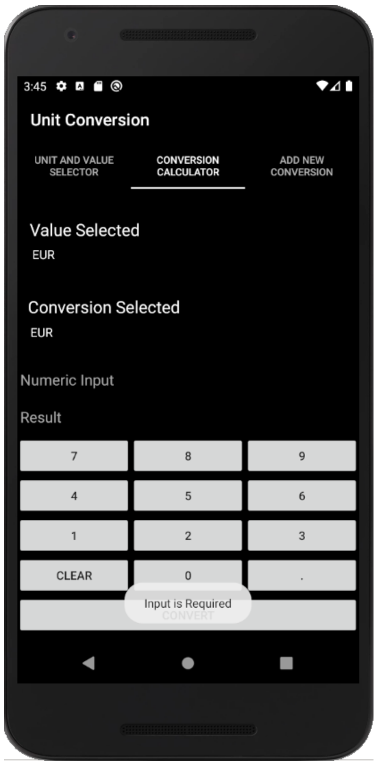
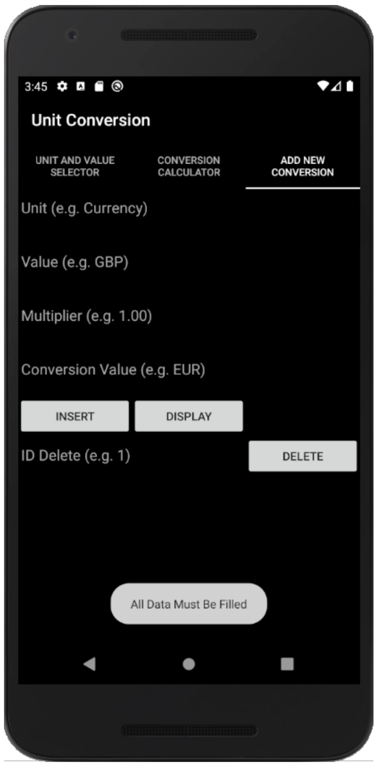


Figure 13.
Page Three Fragment Landscape from the Emulator



Input is Required

Figure 14.
Page Two Fragment Error Message from the Emulator



All Data Must Be Filled

Figure 15.
Page Three Fragment Error Message from the Emulator

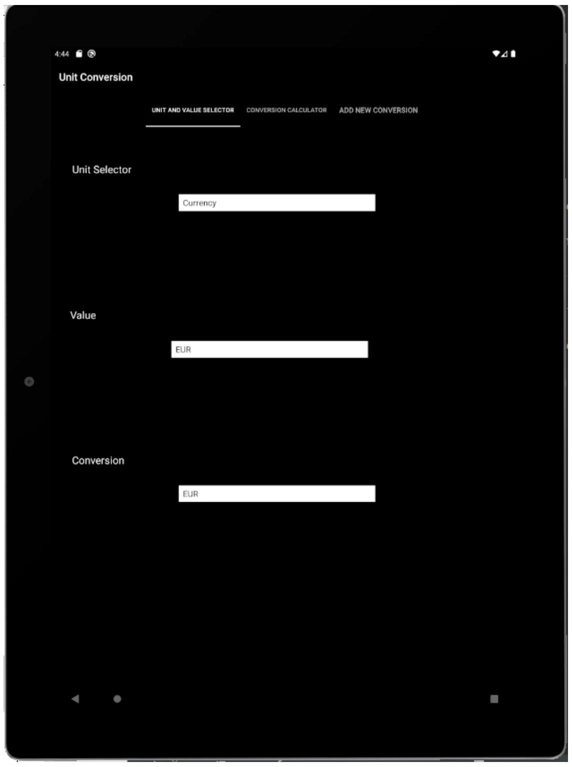


Figure 16.
Page One Fragment on a Tablet from the Emulator

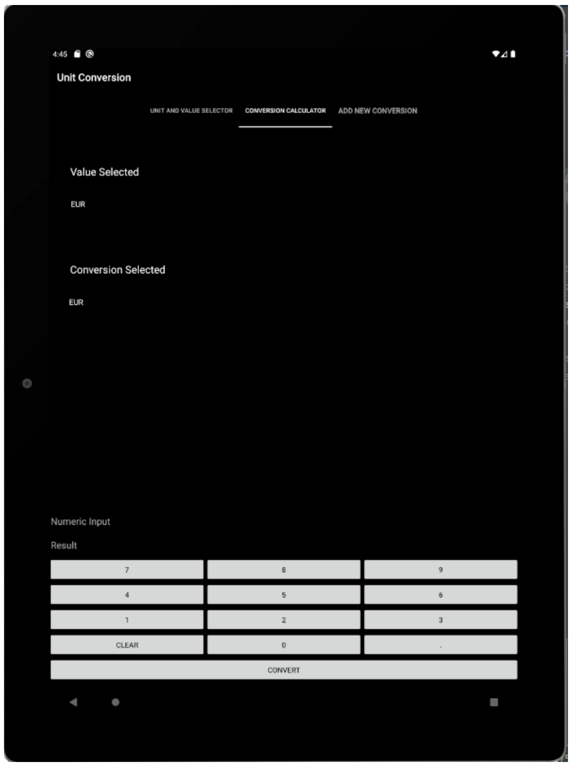


Figure 17.
Page Two Fragment on a Tablet from the Emulator

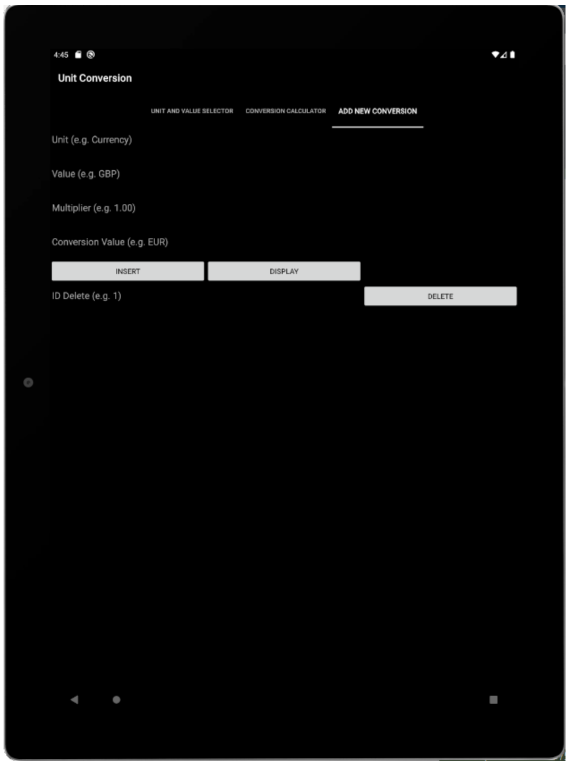


Figure 18.
Page Three Fragment on a Tablet from the Emulator

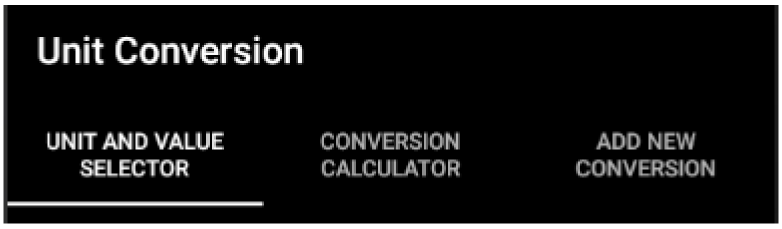


Figure 19.
Navigation Bar

4. Features Implemented

4.1. Basic Features and Functionality

I was able to develop and implement all the basic features and functionality required in the brief. The application supports two fragments that hold the basic features such as the first fragment holding the spinners required for selecting units, values, and conversions. These spinners offer several different fixed values and units that the user can select from. The second fragment contains an inbuilt calculator that allows the user to input numeric values to then be automatically converted when entered. The calculator also contains a clear button, that when pressed, the input and calculated results are cleared from the calculators displays. Also displayed on this fragments page is the two conversion values to show what is being converted from and converted to. I have also managed to get the calculator to auto convert values as they are inputted and displayed onto the screen.

4.2. Advanced Features and Functionality

I was able to implement all features talked about in the press to extend the functionality of the application, for the persistent storage, I decided to develop and implement an SQLite database. Even though there are several different conversion values hardcoded and inserted into the database, the database allows the user to insert any type of conversion they want. They can then use their own created conversions throughout the application as I have made it, so the spinner values are retrieved from the database using array adapters to store and display the data. Having this done meant that the spinners are always up to date with whatever is currently stored in the database. With data being inserted into the database, I had to implement a number of validation checks to ensure that the user did not enter any data that would potentially break the application when a conversion is carried out. There are also validation checks for all text fields throughout the application to check if input has been entered, this was done to prevent any null values being used as a conversion input or being stored into the database along with avoid duplicate values being added. The database can also display all contents stored inside of it and allows the user to scroll through and see all conversion values including their own created ones that they have added.

Another advanced feature I was able to implement was the ability to rotate the screen correctly and have the application adapt the changes. Using the database meant that I was able to keep values displayed in the spinners and text fields to stay the same as they would normally resort back their default values. I was also able to make sure the application works with a bigger screen such as a tablet, the layout changes to handle the change in screen size.

5. Future Improvements

Overall, I am very satisfied with the work that I have been able to produce with the application containing everything required plus extra features and functionality. But there is several different changes and additional features that I would have liked to implement if I had more time or if I were to work on the application in the future. These features and functionality additions are:

Deleting Conversion Values from the Database Correctly

I attempted to implement this feature but was only able to delete a table which deletes everything held in the database which is not a good idea as this would break the application. I did manage to implement the delete button along with required validation, but the button currently does not actually do anything when an id is entered. The delete function would delete a single row in the database using a unique id that user would input.

Automatic Reverse Conversion

Currently when a conversion is added into the database, the user will need enter two sperate entries for a conversion and the reverse conversion. Having the ability to just enter one conversion and have the application create the reverse would save the user having to create a second conversion. This would create a more efficient way to insert data into the database and avoid the risk of forgetting to create a reverse conversion.

Broader Compatibility

Currently the application is only designed for android, it would be good to expand out and have the application compatible with different operating systems such as Windows and iOS.

User Accessibility

The application is set to dark mode with the chosen colour scheme and currently cannot be changed, this could be changed with the ability of a press of a button to change the colour scheme to a selection of colours. Another accessibility addition that be added is the ability for the user to change the font size and potentially the font itself. The final accessibility feature that I could be added is different language conversion support, this would allow the user to change the language displayed on the screen with the use of a selection of languages. These features would provide greater accessibility for the user and create a more friendly user experience.

6. Code Listings

6.1. Main Activity Class

```
package uk.ac.stir.cs.unitconv

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import androidx.viewpager.widget.ViewPager
import com.google.android.material.tabs.TabLayout
import androidx.appcompat.widget.Toolbar

/**
 * This class controls all the applications with the ability to switch between the
 * different fragments.
 */
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        // Sets the activity main xml as the default view when the application is
loaded up
        setContentView(R.layout.activity_main)

        // Finds the tool in the xml file and saves in the toolbar variable
        val toolbar = findViewById<Toolbar>(R.id.toolbar)
        // Passes in the toolbar variable as the second tool on the screen
        setSupportActionBar(toolbar)

        // Sets the names of the of the tabs in the tab layout at tge top of the
screen
        val tabLayout = findViewById<TabLayout>(R.id.tab_layout)
        tabLayout.addTab(tabLayout.newTab().setText(R.string.tab_page1))
        tabLayout.addTab(tabLayout.newTab().setText(R.string.tab_page2))
        tabLayout.addTab(tabLayout.newTab().setText(R.string.tab_page3))
        tabLayout.tabGravity = TabLayout.GRAVITY_FILL

        val viewPager = findViewById<ViewPager>(R.id.pager)
        val adapter = PageAdapter(supportFragmentManager, tabLayout.tabCount)

        viewPager.adapter = adapter

viewPager.addOnPageChangeListener(TabLayout.TabLayoutOnPageChangeListener(tabLayou
t))
        tabLayout.addOnTabSelectedListener(object :
TabLayout.OnTabSelectedListener {
            override fun onTabSelected(tab: TabLayout.Tab) {
                viewPager.currentItem = tab.position
            }

            override fun onTabUnselected(tab: TabLayout.Tab) {}
            override fun onTabReselected(tab: TabLayout.Tab) {}
        })
    }
}
```

6.2. Page Adapter Class

```
package uk.ac.stir.cs.unitconv

import androidx.fragment.app.Fragment
import androidx.fragment.app.FragmentManager
import androidx.fragment.app.FragmentStatePagerAdapter

internal class PageAdapter(fragmentManager: FragmentManager?, private val
mNumOfTabs:
Int) : FragmentStatePagerAdapter(fragmentManager!!,
    BEHAVIOR_RESUME_ONLY_CURRENT_FRAGMENT) {

    /**
     * Gets the position of what fragment is currently being displayed.
     *
     * @return position The current tab that is being displayed
     */
    override fun getItem(position: Int): Fragment {
        return when (position) {
            0 -> Page1Fragment()
            1 -> Page2Fragment()
            2 -> Page3Fragment()
            else -> Fragment()
        }
    }

    /**
     * Gets the number of the tabs that have been created.
     *
     * @return mNumofTabs The number of tabs that have been created and added
     */
    override fun getCount(): Int {
        return mNumOfTabs
    }
}
```

6.3. Page One Fragment Class

```
package uk.ac.stir.cs.unitconv

import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.AdapterView
import android.widget.ArrayAdapter
import android.widget.Spinner
import androidx.fragment.app.Fragment
import androidx.lifecycle.ViewModelProviders

/**
 * This class holds the fragment that controls the spinners that uses the data
stored in
 * the database.
 * Here the user can select the unit and conversion values that they want to use
when they
 * get passed into other fragments in the application.
 */
class Page1Fragment : Fragment() {
    private var unitValue: String = ""
    private var convertFromValue: String = ""
    private var unitCache: Int = 0
    private var valueCache: Int = 0
    private var conversionCache: Int = 0

    override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?,
savedInstanceState: Bundle?): View? {
        return inflater.inflate(R.layout.page1_fragment, container, false)
    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)

        val fragmentViewModel =
ViewModelProviders.of(activity!!).get(FragmentViewModel::class.java)
        var dataBase = UnitDataBase(context!!)

        // Declaring and initialising the spinners by matching them up with the
spinners from the xml files
        val unitSpinner = view.findViewById(R.id.unitSpinner) as Spinner
        val valueSpinner = view.findViewById(R.id.valueSpinner) as Spinner
        val conversionSpinner = view.findViewById(R.id.conversionSpinner) as
Spinner

        // The units array is created as an array adapter with the layouts of
simple spinner items and drop downs
        var unitsArray: ArrayAdapter<String> =
            ArrayAdapter(context, android.R.layout.simple_spinner_item)

unitsArray.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item)

        // The value array is created as an array adapter with the layouts of
simple spinner items and drop downs
        var valueArray: ArrayAdapter<String> =
            ArrayAdapter(context, android.R.layout.simple_spinner_item)
```

```
valueArray.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item)

        // The conversion array is created as an array adapter with the layouts of
simple spinner items and drop downs
        var conversionArray: ArrayAdapter<String> =
            ArrayAdapter(context, android.R.layout.simple_spinner_item)

conversionArray.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_i
tem)

        // Sets the spinners with there relevant arrays
        unitSpinner.adapter = unitsArray
        valueSpinner.adapter = valueArray
        conversionSpinner.adapter = conversionArray

        // Method calls the update all the arrays by passing them in
        updateAllArrays(dataBase, unitsArray, valueArray, conversionArray,
fragmentViewModel)

        // Method calls to carryout the actions for the unit spinner using the the
database, the view model and all the spinners
        unitSpinnerActions(unitSpinner, dataBase, unitsArray, valueArray,
conversionArray, fragmentViewModel)

        // Method calls to carryout the actions for the value spinner using the
the database, the view model and all the spinners
        valueSpinnerActions(valueSpinner, dataBase, unitsArray, valueArray,
conversionArray, fragmentViewModel)

        // Method calls to carryout the actions for the conversion spinner using
the the database, the view model and all the spinners
        conversionSpinnerActions(conversionSpinner, dataBase, unitsArray,
valueArray, conversionArray, fragmentViewModel)
    }

    /**
     * Contains the actions for the unit spinner when a selection has been made,
sets the selection
     * and updates the units cache.
     *
     * @param unitSpinner Passes in the spinner that will be used
     * @param dataBase The instance of the database that will be used
     * @param valueArray The instance of the value array that will be used
     * @param conversionArray The instance of the conversion array that will be
used
     * @param fragmentViewModel The instance of the view model that will be used
     */
    private fun unitSpinnerActions(unitSpinner: Spinner, dataBase: UnitDataBase,
unitsArray: ArrayAdapter<String>, valueArray: ArrayAdapter<String>,
conversionArray: ArrayAdapter<String>, fragmentViewModel: FragmentViewModel) {
        unitSpinner.onItemSelectedListener = object :
AdapterView.OnItemSelectedListener {
            override fun onNothingSelected(parent: AdapterView<*>?) {
            }

            override fun onItemSelected(parent: AdapterView<*>, view: View,
position: Int, id: Long) {
                // Stores the position of the value that has been selected in the
spinner as the units cache
```

```
                // This makes sure that the spinner and displays the same when a
changed has been made with the application
                unitCache = position
                // Method call ensures the all the arrays in the fragment are
always up to date when changes are made
                updateAllArrays(dataBase, unitsArray, valueArray, conversionArray,
fragmentViewModel)
                // Sets the value that is stored in the cache into the spinner
display
                unitSpinner.setSelection(unitCache)
            }
        }
    }

    /**
     * Contains the actions for the value spinner when a selection has been made,
sets the selection
     * and updates the value cache.
     *
     * @param valueSpinner Passes in the spinner that will be used
     * @param dataBase The instance of the database that will be used
     * @param valueArray The instance of the value array that will be used
     * @param conversionArray The instance of the conversion array that will be
used
     * @param fragmentViewModel The instance of the view model that will be used
     */
    private fun valueSpinnerActions(valueSpinner: Spinner, dataBase: UnitDataBase,
unitsArray: ArrayAdapter<String>, valueArray: ArrayAdapter<String>,
conversionArray: ArrayAdapter<String>, fragmentViewModel: FragmentViewModel) {
        valueSpinner.onItemSelectedListener = object :
AdapterView.OnItemSelectedListener {
            override fun onNothingSelected(parent: AdapterView<*>?) {
            }

            override fun onItemSelected(parent: AdapterView<*>, view: View,
position: Int, id: Long) {
                // Stores the position of the value that has been selected in the
spinner as the value cache
                // This makes sure that the spinner and displays the same when a
changed has been made with the application
                valueCache = position
                // Method call ensures the all the arrays in the fragment are
always up to date when changes are made
                updateAllArrays(dataBase, unitsArray, valueArray, conversionArray,
fragmentViewModel)
                // Sets the value that is stored in the cache into the spinner
display
                valueSpinner.setSelection(valueCache)
            }
        }
    }

    /**
     * Contains the actions for the conversion spinner when a selection has been
made, sets the selection
     * and updates the conversion cache.
     *
     * @param conversionSpinner Passes in the spinner that will be used
     * @param dataBase The instance of the database that will be used
     * @param valueArray The instance of the value array that will be used
     * @param conversionArray The instance of the conversion array that will be
used
     * @param fragmentViewModel The instance of the view model that will be used
     */
    private fun conversionSpinnerActions(conversionSpinner: Spinner, dataBase:
UnitDataBase, unitsArray: ArrayAdapter<String>, valueArray: ArrayAdapter<String>,
conversionArray: ArrayAdapter<String>, fragmentViewModel: FragmentViewModel) {
        conversionSpinner.onItemSelectedListener = object :
AdapterView.OnItemSelectedListener {
            override fun onNothingSelected(parent: AdapterView<*>?) {
            }

            override fun onItemSelected(parent: AdapterView<*>, view: View,
position: Int, id: Long) {
                // Stores the position of the value that has been selected in the
spinner as the conversion cache
                // This makes sure that the spinner and displays the same when a
changed has been made with the application
                conversionCache = position
                // Method call ensures the all the arrays in the fragment are
always up to date when changes are made
                updateAllArrays(dataBase, unitsArray, valueArray, conversionArray,
fragmentViewModel)
                // Sets the value that is stored in the cache into the spinner
display
                conversionSpinner.setSelection(conversionCache)
            }
        }
    }

    /**
     * Updates all the arrays to ensure all the data the is being held within the
spinners are the most up to date.
     * It also updates the values for the other fragments for the values that
haven been selected.
     *
     * @param dataBase The instance of the database that will be used
     * @param valueArray The instance of the value array that will be used
     * @param conversionArray The instance of the conversion array that will be
used
     * @param fragmentViewModel The instance of the view model that will be used
     */
    private fun updateAllArrays(dataBase: UnitDataBase, unitsArray:
ArrayAdapter<String>, valueArray: ArrayAdapter<String>, conversionArray:
ArrayAdapter<String>, fragmentViewModel: FragmentViewModel) {
        // Method called to clear the arrays each time to enable them to be the
most up to date
        clearArrays(unitsArray, valueArray, conversionArray)

        // Method called to update the the value that has been selected and
returns a value list which gets stored in the spinnerList
        var spinnerList = updateUnitValue(dataBase)
        // For statement used to go through the spinnerList and adds each value
into the unitsArray
        for (i in spinnerList) {
            unitsArray.add(i)
        }
        // Method called to update the the value that has been selected and
returns a conversion list which gets stored in the spinnerList
        spinnerList = updateConversionValue(dataBase)
        // For statement used to go through the spinnerList and adds each value in
the unitsArray
        for (i in spinnerList) {
            valueArray.add(i)
        }

        // Method call to get the display of the unitValue and the
convertFromValue, this gets stored in the conversionList
        var conversionList = dataBase.displayData(unitValue + convertFromValue)

        // If statement used to check if the conversions cache value if greater
than the conversions list size
        // If this is true, the conversions cache value gets set to 0
        if (conversionCache > conversionList.size) {
            conversionCache = 0
        }

        // For statement used to go through the conversionList and adds each value
into the conversionArray
        for (i in conversionList) {
            conversionArray.add(i.convertTo)
        }

        // Method calls for notifying the database that changes have been made to
each array
        unitsArray.notifyDataSetChanged()
        valueArray.notifyDataSetChanged()
        conversionArray.notifyDataSetChanged()

        // The fragmentViewModel is used to communicate to other fragments, this
allows the other
        // fragments use the values that have been selected from this fragment
        // The conversion values will be able to be used for displaying in the
other fragments
        fragmentViewModel.conversion.value = conversionList[conversionCache]
    }

    /**
     * Clears all of the arrays.
     *
     * @param unitsArray the reference being passed in for the unit array
     * @param valueArray the reference being passed in for the value array
     * @param conversionArray the reference being passed in for the conversion
array
     */
    private fun clearArrays(unitsArray: ArrayAdapter<String>, valueArray:
ArrayAdapter<String>, conversionArray: ArrayAdapter<String>) {
        unitsArray.clear()
        valueArray.clear()
        conversionArray.clear()
    }

    /**
     * Updates the unit values that has been selected along with the cache value.
     * Sets the unitValue to the value in the database to be displayed in the
spinner.
```

```
     *
     * @param dataBase The instance of the database that will be used
     *
     * @return valueList Returns a mutable list of the value
     */
    private fun updateUnitValue(dataBase: UnitDataBase): MutableList<String> {
        var valueList = dataBase.spinnerInfo(" GROUP BY category", "category")

        // If statement used to check if the value list is currently empty
        if (valueList.isNotEmpty()) {
            // If statement used to check if the current value cache is greater
than the value lists size
            if (valueCache > valueList.size) {
                // If the value cache is greater than the list size the cache
value is set to 0
                valueCache = 0
            }
            // If the list is empty, the unitValue is set to value in the database
using the unit cache to find the required value
            unitValue = " WHERE category = \'" + valueList[unitCache] + "\'"
        } else {
            // Else statement used to set the unitValue to nothing if the value
list is not empty
            unitValue = ""
        }

        // Returns the valueList which will contain the value selected by the user
        return valueList
    }

    /**
     * Updates the conversion values that has been selected along with the
conversion value.
     * Sets the conversionValue to the value in the database to be displayed in
the spinner.
     *
     * @param dataBase The instance of the database that will be used
     *
     * @return conversionList Returns a mutable list of the value
     */
    private fun updateConversionValue(dataBase: UnitDataBase): MutableList<String>
{
        var conversionList = dataBase.spinnerInfo("$unitValue GROUP by
convertFrom", "convertFrom")

        // If statement used to check if the conversion list is currently empty
        if (conversionList.isNotEmpty()) {
            // If statement used to check if the current conversion cache is
greater than the conversion lists size
            if (conversionCache > conversionList.size) {
                // If the value cache is greater than the list size the cache
value is set to 0
                conversionCache = 0
            }
            // If the list is empty, the convertFromValue is set to value in the
database using the conversion value cache to find the required conversion
            convertFromValue = " convertFrom = \'" + conversionList[valueCache] +
"\'"
```

```
            // If statement used and stored in convertFromValue to check is the
unitValue is not empty
            convertFromValue = if (unitValue.isNotEmpty()) {
                // If unit value is not empty the AND query is added to the
convertFromValue
                " AND$convertFromValue"
            } else {
                // Else if the unit value is empty the WHERE query is added to the
convertFromValue
                " WHERE$convertFromValue"
            }
            // Else statement used to set the convertFromValue to nothings if the
value list is not empty
        } else {
            convertFromValue = ""
        }

        // Returns the conversionList which will contain the conversion selected
by the user
        return conversionList
    }
}
```

6.4. Page Two Fragment Class

```
package uk.ac.stir.cs.unitconv

import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.Button
import android.widget.EditText
import android.widget.Toast
import androidx.fragment.app.Fragment
import androidx.lifecycle.Observer
import androidx.lifecycle.ViewModelProviders
import kotlinx.android.synthetic.main.page2_fragment.*
import java.lang.Exception
import java.text.DecimalFormat

/**
 * This class contains the fragment that holds the conversion calculator, this is
where
 * the user can enter the number that they want to convert.
 */
class Page2Fragment : Fragment() {

    private var unitValue: String = ""
    private var conversionValue: String = ""

    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        return inflater.inflate(R.layout.page2_fragment, container, false)
    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)

        val fragmentViewModel =
ViewModelProviders.of(activity!!).get(FragmentViewModel::class.java)
        val dataBase = UnitDataBase(context!!)
        var conversionData: ConversionData = ConversionData()

        // The fragmentVIewModel allows communication between the two fragments,
allows data to be passed through
        // The observe is used as a listener for the data that been sent from
other fragments, this is sent when the user
        // selects the values they want to convert to and from
        fragmentViewModel.conversion.observe(this, Observer<Any> { conversion ->
            conversionData = conversion as ConversionData

            // Sets the text views to the values that have been selected
            selectedValueDisplayTextView.text = conversionData.convertFrom
            selectedConversionDisplayTextView.text = conversionData.convertTo

            // Sets the values to the the values that have been selected
            unitValue = conversionData.convertFrom
            conversionValue = conversionData.convertTo
        })
        /**
         * Initialises all required text edits and buttons used in the
application.
         */
        val calculationInput = view.findViewById(R.id.calculationInputTextView) as
EditText
        val calculationResult = view.findViewById(R.id.calculationResultTextView)
as EditText

        val calculatorButtonNumber0 =
view.findViewById(R.id.calculatorButtonNumber0) as Button
        val calculatorButtonNumber1 =
view.findViewById(R.id.calculatorButtonNumber1) as Button
        val calculatorButtonNumber2 =
view.findViewById(R.id.calculatorButtonNumber2) as Button
        val calculatorButtonNumber3 =
view.findViewById(R.id.calculatorButtonNumber3) as Button
        val calculatorButtonNumber4 =
view.findViewById(R.id.calculatorButtonNumber4) as Button
        val calculatorButtonNumber5 =
view.findViewById(R.id.calculatorButtonNumber5) as Button
        val calculatorButtonNumber6 =
view.findViewById(R.id.calculatorButtonNumber6) as Button
        val calculatorButtonNumber7 =
view.findViewById(R.id.calculatorButtonNumber7) as Button
        val calculatorButtonNumber8 =
view.findViewById(R.id.calculatorButtonNumber8) as Button
        val calculatorButtonNumber9 =
view.findViewById(R.id.calculatorButtonNumber9) as Button
        val calculatorButtonDot = view.findViewById(R.id.calculatorButtonDot) as
Button

        val calculatorButtonClear = view.findViewById(R.id.calculatorButtonClear)
as Button
        val calculatorButtonEnter = view.findViewById(R.id.calculatorButtonEnter)
as Button

        /**
         * Sets the inputted values into the calculationInput text area.
         */
        fun setValue(value: Int) {
            calculationInput.setText(

StringBuilder().append(calculationInput.text.toString()).append(value).toString()
            )
        }

        /**
         * Clears the input from the calculationsInput text area and the results
from the calculationResult area.
         */
        fun clearValues() {
            calculationInput.text = null
            calculationResult.text = null
        }

        /**
         * Converts the calculation result into text and formats it correctly to
```

```
be displayed.
         */
        fun inputConversion() {
            // Gets the conversion rate from the database using the two values
selected and stores the returned value into conversionRate
            val conversionRate: Double =
                dataBase.conversionRate(dataBase.readableDatabase, unitValue,
conversionValue)
            // Sets the format for decimal values
            val valueFormat = DecimalFormat("#,###.#######")

            // Try catch used to ensure that input has been entered into the
calculation field so the application does not crash when no value has been entered
            try {
                // Sets the calculation result as text along with the value name
to be displayed in the calculation result field
                calculationResult.setText(
                    java.lang.StringBuilder().append(
                        valueFormat.format(
                            conversionRate.times(
                                calculationInput.text.toString().toDouble()
                            )
                        )
                    ).append(" $conversionValue")
                )
                // Catches any exception that are thrown
            } catch (e: Exception) {
                // Message displayed to the screen is there is no input enetered
                Toast.makeText(context, "Input is Required",
Toast.LENGTH_SHORT).show()
            }
        }

        // Sets the commands for all the button included in the calculator,
        // sets each numbered button along with enter and clear processes
        calculatorButtonNumber0.setOnClickListener {
            setValue(0)
        }

        calculatorButtonNumber1.setOnClickListener {
            setValue(1)
        }

        calculatorButtonNumber2.setOnClickListener {
            setValue(2)
        }

        calculatorButtonNumber3.setOnClickListener {
            setValue(3)
        }

        calculatorButtonNumber4.setOnClickListener {
            setValue(4)
        }

        calculatorButtonNumber5.setOnClickListener {
            setValue(5)
        }
        calculatorButtonNumber6.setOnClickListener {
            setValue(6)
        }

        calculatorButtonNumber7.setOnClickListener {
            setValue(7)
        }

        calculatorButtonNumber8.setOnClickListener {
            setValue(8)
        }

        calculatorButtonNumber9.setOnClickListener {
            setValue(9)
        }

        // Couldn't get this to work correctly so not fully implemented
        calculatorButtonDot.setOnClickListener {

        }

        calculatorButtonEnter.setOnClickListener {
            inputConversion()
        }

        calculatorButtonClear.setOnClickListener {
            clearValues()
        }
    }
}
```

6.5. Page Three Fragment Class

```kotlin
package uk.ac.stir.cs.unitconv

import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.Toast
import androidx.fragment.app.Fragment
import kotlinx.android.synthetic.main.page3_fragment.*

/**
 * This class contains the fragment that controls the ability to insert, display
and
 * delete conversion values from the database.
 */
class Page3Fragment : Fragment() {

    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        return inflater.inflate(R.layout.page3_fragment, container, false)
    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)

        val dataBase = UnitDataBase(context!!)

        // Method called to add the default data into the database
        addDefaultData(dataBase)
        // Method called for in the insertion of data in the database from the
user
        insertData(dataBase)
        // Method called for displaying the data stored in the database
        displayData(dataBase)
        // Method called for deleting data from the database
        deleteData(dataBase)
    }

    /**
     * Adds the default data into the database, these values will always
     * be present in the database and cannot be deleted.
     *
     * @param dataBase The instance of the database that will be used
     */
    private fun addDefaultData(dataBase: UnitDataBase) {
        var displayData = dataBase.displayData("")

        // Checks to see if the default data is already inserted to prevent
duplication.
        // If there is no data, the default data gets added.
        if (displayData.isEmpty()) {
            // Inserts default values that are always in the database
            dataBase.insertDefaultValues()
        }
    }
    /**
     * Enables the user to enter in their own values for a conversion and
     * insert it into the database to then be used throughout the application.
     *
     * @param dataBase The instance of the database that will be used
     */
    private fun insertData(dataBase: UnitDataBase) {
        var conversionData: ConversionData

        var unit: String
        var value: String
        var multiplier = 0.0
        var conversion: String

        // This button performs the action of inserting a new conversion into the
database that the user has entered
        insertButton.setOnClickListener {
            // Converts the inputs from the texts fields into Strings and makes
sure that the first character of the string is a capital
            // and the rest of the characters are lower case. They then get stored
into their relevant variable.
            unit = newUnitInputEditText.text.toString().toLowerCase().capitalize()
            value =
newValueInputEditText.text.toString().toLowerCase().capitalize()
            conversion =
newConversionInputEditText.text.toString().toLowerCase().capitalize()

            // If statement used to check is the multiplier input is empty
            if (newMultiplierInputEditText.text.toString().isNotEmpty()) {
                // Converts the input from the text fields into a double and
stores the value into the multiplier variable
                multiplier = newMultiplierInputEditText.text.toString().toDouble()
            }

            // If statement used to validate the entered values
            // If valid, the data is inserted into the database
            // IF invalid, the relevant validation check is carried out
            if (validateInput(unit, value, multiplier, conversion)) {
                // Collects all the entered data into on variable
                conversionData = ConversionData(unit, value, multiplier,
conversion)

                // Inserts the new data conversion into the database
                dataBase.insertData(conversionData)
                // Displays the database on the screen
                displayButton.performClick()
                // Calls the clear text method to clear the text fields
                clearText()
            }
        }
    }

    /**
     * Clears all the text fields that are displayed on the screen.
     */
    private fun clearText() {
        newUnitInputEditText.setText("")
        newValueInputEditText.setText("")
        newMultiplierInputEditText.setText("")
        newConversionInputEditText.setText("")
        deleteDatabaseValueNumberInput.setText("")
    }

    /**
     * Validation is carried out on the inputted values before it gets added into
the database.
     *
     * @param unit Passes in the unit entered by the user
     * @param value Passes in the values entered by the user
     * @param multiplier Passes in the multiplier entered by the user
     * @param conversion Passes in the conversion entered by the user
     *
     * @return inputIsValid Returns either true or false
     */
    private fun validateInput(
        unit: String,
        value: String,
        multiplier: Double,
        conversion: String
    ): Boolean {
        var inputIsValid = true

        // Regex checks the text fields for for any letter in either lowercase,
uppercase and if it contains spaces
        val textRegex = "^[a-zA-Z ]+$".toRegex()
        // Regex checks the text field for any numbers and will allow one decimal
space if required
        val decimalRegex = "^\\d{0,9}\\.\\d{1,4}$".toRegex()
        // Regex checks the text field for any whole numbers
        val integerRegex = "^[0-9]*$".toRegex()

        // If statement used to check all input fields to ensure that none are
empty
        if (unit == "" || value == "" || multiplier == 0.0 || conversion == "") {
            // Displays a message to the screen if at least on field is empty
            Toast.makeText(context, "All Data Must Be Filled",
Toast.LENGTH_SHORT).show()
            inputIsValid = false
        }

        // If statement used to ensure that the value and conversion that has been
entered is not the same
        else if (value == conversion || conversion == value) {
            Toast.makeText(context, "Value and Conversion Cannot be the Same",
Toast.LENGTH_SHORT)
                .show()
            inputIsValid = false
        }

        // If statement used to check the if the unit only contains letters
        else if (!unit.matches(textRegex)) {
            // Displays a message to the screen if a non text character is entered
for the unit
            Toast.makeText(context, "Unit can Only Contain Letters",
Toast.LENGTH_SHORT).show()
            inputIsValid = false
        }
        // If statement used to check if the value only contains letters
        else if (!value.matches(textRegex)) {
            // Displays a message to the screen if a non text character is entered
for the value
            Toast.makeText(context, "Value can Only Contain Letters",
Toast.LENGTH_SHORT).show()
            inputIsValid = false
        }

        // If statement used to check if the conversion only contains letters
        else if (!conversion.matches(textRegex)) {
            // Displays a message to the screen if a non text character is entered
for the conversion
            Toast.makeText(context, "Conversion can Only Contain Letters",
Toast.LENGTH_SHORT)
                .show()
            inputIsValid = false
        }

        // Unable to get this validation working correctly

//        // If statement used to check if the multiplier only contains whole or
decimal numbers
//        else if (!multiplier.toString().matches(integerRegex) ||
!multiplier.toString().matches(decimalRegex)) {
//            // Displays a message to the screen if a non integer or decimal
character is entered
//            Toast.makeText(context, "Conversion can Only Contain Whole Numbers
or Decimals", Toast.LENGTH_SHORT).show()
//            inputIsValid = false
//        }

        // Returns either true or false
        return inputIsValid
    }

    /**
     * Displays all the data that is currently being stored in the database.
     *
     * @param dataBase The instance of the database that will be used
     */
    private fun displayData(dataBase: UnitDataBase) {
        // This button performs the action of displaying the contents of the
database
        displayButton.setOnClickListener() {
            var displayData = dataBase.displayData("")

            dataDisplayTextView.text = ""

            // If else statement used to check is displayData is empty
            if (displayData.isEmpty()) {
                // Displays a message to the screen telling the user that the
database is currenlty empty
                Toast.makeText(context, "Database Currently Empty",
Toast.LENGTH_SHORT).show()
            } else {
                // For statement used to go through every value in the database to
then create string to be displayed to the screen
                for (i in 0 until displayData.size) {
                    dataDisplayTextView.append(
                        displayData[i].id.toString() + ". " + "Unit: " +
displayData[i].category + " Value: " + displayData[i].convertFrom + " Multiplier:
" +
                                displayData[i].multiplier + " Conversion: " +
displayData[i].convertTo + "\n"
                    )
                }
            }
        }
    }

    /**
     * This currently does not work correctly as it does not find the id to be
deleted.
     *
     * Deletes entities from the database with the use of entering the specific
id.
     *
     * @param dataBase
     */
    private fun deleteData(dataBase: UnitDataBase) {
        var displayData = dataBase.displayData("")
        var id: Int = 0

        val dataBase = UnitDataBase(context!!)

        var conversionData = ConversionData()

        // This button performs the action of deleting a selected id from the
database
        deleteButton.setOnClickListener() {
            // If statement used to check if has been any input entered
            if (deleteDatabaseValueNumberInput.text.toString().isNotEmpty()) {
                // Sets the id variable value to the number that has been entered
by the user in the delete number field
                id = deleteDatabaseValueNumberInput.text.toString().toInt()

                // If statement used to check if the id entered is not a default
value
                if (id > 80) {
                    // If statement used to check if the id entered is the same as
the saved id in the conversionData
                    if (id == conversionData.id) {
                        // Calls the delete method in the database class with the
id to be deleted passed in
                        dataBase.deleteData(id)
                    }

                    // Displays a message to the screen
                    Toast.makeText(context, "Successfully Deleted",
Toast.LENGTH_SHORT).show()
                    // Displays the database to the screen
                    displayButton.performClick()
                    // Calls the clear text method to clear the text fields
                    clearText()

                    // Else if statement used to check if the id is a default
value
```

```
                } else if (id > 0 || id <= 80) {
                    // Message displayed to screen stating that default values
cannot be deleted from the database
                    Toast.makeText(context, "Cannot Delete Default Values",
Toast.LENGTH_SHORT)
                        .show()
                }
                // Else if statement used to check is the database is currently
empty
            } else if (displayData.isEmpty()) {
                // Message displayed to the screen telling the user that the
database is currently empty
                Toast.makeText(context, "Database Currently Empty",
Toast.LENGTH_SHORT).show()
            }
        }
    }
}
```

6.6. Conversion Data Class

```
package uk.ac.stir.cs.unitconv

/**
 * This class holds all the required data for the data base,
 * it uses the data form the inputted values from the user and
 * adds it as anew entry into the UnitDataBase.
 *
 */
class ConversionData {
    var id: Int = 0
    var category: String = ""
    var convertFrom: String = ""
    var multiplier: Double = 0.0
    var convertTo: String = ""

    /**
     * The constructor is used to initialize the objects that will be
     * used for the database.
     */
    constructor(category: String, convertFrom: String, multiplier: Double,
convertTo: String) {
        this.category = category
        this.convertFrom = convertFrom
        this.multiplier = multiplier
        this.convertTo = convertTo
    }

    // Runs the constructor
    constructor()
}
```

6.7. Fragment View Model Class

```
package uk.ac.stir.cs.unitconv

import androidx.lifecycle.MutableLiveData
import androidx.lifecycle.ViewModel

/**
 * This class is used to create connections to be used between the two page
fragments,
 * this allows data to be passed on when selected from the spinner.
 */
class FragmentViewModel : ViewModel() {
    val conversion: MutableLiveData<ConversionData> by lazy {
        MutableLiveData<ConversionData>()
    }
}
```

6.8. Unit Database Class

```kotlin
package uk.ac.stir.cs.unitconv

import android.content.ContentValues
import android.content.Context
import android.database.sqlite.SQLiteDatabase
import android.database.sqlite.SQLiteOpenHelper
import android.widget.Toast

// Declaration and initialising of database values
const val DATABASE_NAME = "conversionDataBase"
const val TABLE_NAME = "conversionInfo"
const val COLUMN_ID = "id"
const val COLUMN_CATEGORY = "category"
const val COLUMN_CONVERT_FROM = "convertFrom"
const val COLUMN_MULTIPLIER = "multiplier"
const val COLUMN_CONVERT_TO = "convertTo"

/**
 * The database class is used to hold all of the data required for the
functionality of the application.
 * Enables the user to input and remove conversions for them to then use.
 *
 * @param context Passes in the applications environment
 */
class UnitDataBase(private var context: Context) :
    SQLiteOpenHelper(context, DATABASE_NAME, null, 1) {
    override fun onCreate(dataBase: SQLiteDatabase?) {
        val createTable = "CREATE TABLE " + TABLE_NAME + " (" +
                COLUMN_ID + " INTEGER PRIMARY KEY AUTOINCREMENT," +
                COLUMN_CATEGORY + " VARCHAR(256)," +
                COLUMN_CONVERT_FROM + " VARCHAR(256)," +
                COLUMN_MULTIPLIER + " FLOAT," +
                COLUMN_CONVERT_TO + " VARCHAR(256))"
        dataBase?.execSQL(createTable)
    }

    /**
     * Updates the database by dropping the old table and creates a new table.
     *
     * @param dataBase The instance of the database that will be used
     * @param oldVersion Passes in the current version of the database
     * @param newVersion Passes in the new version of the database
     */
    override fun onUpgrade(dataBase: SQLiteDatabase, oldVersion: Int, newVersion:
Int) {
        dataBase.execSQL("DROP TABLE IF EXISTS $TABLE_NAME")
        // Calls the onCreate method to make the new version of the database
        onCreate(dataBase)
    }

    /**
     * This method controls how data gets inserted into the database, it ensures
that all
     * the data getting passed in is valid.
     * Inserted data can be used to carryout conversions.
     *
     * @param conversionData The instance of the ConversionData class that will be
used
```

```
     */
    fun insertData(conversionData: ConversionData) {
        // Declaring and initialising the variable that will store the value into
the database
        var contentValues = ContentValues()

        val dataBase = this.writableDatabase

        // Stores all the values stored in the conversionData variables in the
contentVales
        contentValues.put(COLUMN_CATEGORY, conversionData.category)
        contentValues.put(COLUMN_CONVERT_FROM, conversionData.convertFrom)
        contentValues.put(COLUMN_MULTIPLIER, conversionData.multiplier)
        contentValues.put(COLUMN_CONVERT_TO, conversionData.convertTo)

        // Creates an insert statement that adds the new created conversion into
the database
        var input = dataBase.insert(TABLE_NAME, null, contentValues)

        // If statement used to check if the input is valid and can be inserted
into the database
        if (input == (-1).toLong()) {
            // Displays a message to screen if the data was not inserted into the
database
            Toast.makeText(context, "Failed to Insert", Toast.LENGTH_SHORT).show()
        } else {
            // Displays message to the screen if the data was inserted into the
database
            Toast.makeText(context, "Successfully Inserted",
Toast.LENGTH_SHORT).show()
        }
    }

    /**
     * Reads the data that is currently being stored in the database, allows
     * the user to display the data stored with the use of a parameter if
required.
     *
     * @param entry Passes in any required parameters for the query search
     *
     * @return conversionDataList Holds the data retrieved from the database
     */
    fun displayData(entry: String): MutableList<ConversionData> {
        // Mutable List uad as a generic ordered collection of elements that
supports adding and removing elements
        var conversionDataList: MutableList<ConversionData> = ArrayList()

        val dataBase = this.readableDatabase
        // Query created to query and search the database
        val query = "Select * from $TABLE_NAME$entry"
        // The database getting queried using the value stored in query and then
stored into the output variable
        val output = dataBase.rawQuery(query, null)

        // If statement used to ensure that the data stored in output is moved to
the first position
        if (output.moveToFirst()) {
            // Do while statement used to ensure all data gets collected and
stored
```

```
            do {
                var conversionData = ConversionData()

                // The conversionData holds the data held in each index column in
the database by using their column names
                conversionData.id =
output.getString(output.getColumnIndex(COLUMN_ID)).toInt()
                conversionData.category =
output.getString(output.getColumnIndex(COLUMN_CATEGORY))
                conversionData.convertFrom =
                    output.getString(output.getColumnIndex(COLUMN_CONVERT_FROM))
                conversionData.multiplier =

output.getString(output.getColumnIndex(COLUMN_MULTIPLIER)).toDouble()
                conversionData.convertTo =
                    output.getString(output.getColumnIndex(COLUMN_CONVERT_TO))

                // Adds all the data stored in conversionData into the
conversionDataList
                conversionDataList.add(conversionData)
                // Ends the do while loops when the output has moved on the next
output
                // End when all the data has been collected
            } while (output.moveToNext())
        }

        // Closes the query call for the database
        output.close()
        // Closes the connection to the database
        dataBase.close()

        // Returns the conversionDataList that contains all required data
collected
        return conversionDataList
    }

    /**
     * Retrieves all required data for each spinners throughout the application.
     *
     * @param entry Passes in the query entry for the spinner
     * @param unit Passes in the unit to be used for the spinner
     */
    fun spinnerInfo(entry: String, unit: String): MutableList<String> {
        var infoList: MutableList<String> = ArrayList()

        val dataBase = this.readableDatabase
        // Query created to query and search the database
        val query = "Select * from $TABLE_NAME$entry"
        // The database getting queried using the value stored in query and then
stored into the output variable
        val output = dataBase.rawQuery(query, null)

        // If statement used to ensure that the data stored in output is moved to
the first position
        if (output.moveToFirst()) {
            // Do while statement used to ensure all data stored in the infoList
gets added into the relevent spinner
            do {
                infoList.add(output.getString(output.getColumnIndex(unit)))
                // Ends the do while loops when all data has been added into the
spinner
            } while (output.moveToNext())
        }

        // Closes the query call for the database
        output.close()
        // Closes the connection to the database
        dataBase.close()

        // Returns the infoList that contains all the required data for the
spinner
        return infoList
    }


    /**
     * Deletes data from the data base and resets the
     * autoincrement id for the next insert.
     *
     * @param id Passes in the id value requested to be deleted from the database
     */
    fun deleteData(id: Int) {
        val dataBase = this.writableDatabase

        // Deletes requested id from the database
//          dataBase.execSQL("DELETE FROM " + "TABLE_NAME" + " WHERE " + COLUMN_ID +
"=\"" + id + "\";")

        // Deletes the whole table
        dataBase.execSQL("DELETE FROM " + TABLE_NAME)//delete data from table
        // Reset the autoincrement id
        dataBase.execSQL("DELETE FROM sqlite_sequence WHERE NAME = \'" +
TABLE_NAME + "\' ")

        // Closes the connection to the database
        dataBase.close()
    }

    /**
     * Carries out the conversion rate between two different selected values.
     *
     * @param dataBase The instance of the database that will be used
     * @param value The value getting used to convert from
     * @param conversion The conversion value getting used to convert to
     *
     * @return conversionRate Returns the conversion rate between the two
different values entered
     */
    fun conversionRate(dataBase: SQLiteDatabase, value: String, conversion:
String): Double {
        var conversionRate = 0.0

        // Query created to query and search the database
        val query =
            "SELECT $COLUMN_MULTIPLIER FROM $TABLE_NAME WHERE $COLUMN_CONVERT_FROM
== '$value' AND $COLUMN_CONVERT_TO == '$conversion';"
        // The database getting queried using the value stored in query and then
stored into the output variable
```

```
        val output = dataBase?.rawQuery(query, null)

        // If statement used to check that the output variable is not empty
        if (output != null) {
            // If statement used to ensure that the data stored in output is moved
to the first position
            if (output.moveToFirst()) {
                // Gets the multiplier for the conversion between the two selected
values and stores it in the conversionRate variable
                conversionRate =
output.getDouble(output.getColumnIndex(COLUMN_MULTIPLIER))
            }
        }

        // Returns the conversion rate multiplier between the two selected values
        return conversionRate
    }

    /**
     * Inserts the default values into the data base, these vaules will always be
     * present in the database and cannot be deleted.
     */
    fun insertDefaultValues() {
        val dataBase = this.writableDatabase
        var contentValues = ContentValues()

        // Calls a method for each of the values held for the required database
data.
        // Passes in and uses the contentValues and dataBase values to insert to
the data into the database.
        // Method calls for the Currency units.
        conversionInfoGBP(dataBase, contentValues)
        conversionInfoUSD(dataBase, contentValues)
        conversionInfoEUR(dataBase, contentValues)
        conversionInfoJPY(dataBase, contentValues)

        // Method calls for the Measurement units.
        conversionInfoCentimeters(dataBase, contentValues)
        conversionInfoInches(dataBase, contentValues)
        conversionInfoMillimeters(dataBase, contentValues)
        conversionInfoFoot(dataBase, contentValues)

        // Method calls for the Weight units
        conversionInfoStone(dataBase, contentValues)
        conversionInfoPounds(dataBase, contentValues)
        conversionInfoGrams(dataBase, contentValues)
        conversionInfoKilograms(dataBase, contentValues)

        // Method calls for the Volume units
        conversionInfoLiters(dataBase, contentValues)
        conversionInfoMillilitres(dataBase, contentValues)
        conversionInfoPints(dataBase, contentValues)
        conversionInfoGallons(dataBase, contentValues)

        //Method calls for the Speed units
        conversionInfoMPH(dataBase, contentValues)
        conversionInfoKPH(dataBase, contentValues)
        conversionInfoKnot(dataBase, contentValues)
        conversionInfoMPS(dataBase, contentValues)
    }

    /**
     * Conversion data for converting GBP into other currencies.
     *
     * @param dataBase the instance of the database that will be used
     * @param contentValues enables the data to be inserted into the database
     */
    private fun conversionInfoGBP(dataBase: SQLiteDatabase, contentValues:
ContentValues) {
        // Conversion for GBP to GBP
        contentValues.put(COLUMN_CATEGORY, "Currency")
        contentValues.put(COLUMN_CONVERT_FROM, "GBP")
        contentValues.put(COLUMN_CONVERT_TO, "GBP")
        contentValues.put(COLUMN_MULTIPLIER, "1.00")
        dataBase.insert(TABLE_NAME, null, contentValues)

        // Conversion for GBP to USD
        contentValues.put(COLUMN_CATEGORY, "Currency")
        contentValues.put(COLUMN_CONVERT_FROM, "GBP")
        contentValues.put(COLUMN_CONVERT_TO, "USD")
        contentValues.put(COLUMN_MULTIPLIER, "1.33")
        dataBase.insert(TABLE_NAME, null, contentValues)

        // Conversion for GBP to EUR
        contentValues.put(COLUMN_CATEGORY, "Currency")
        contentValues.put(COLUMN_CONVERT_FROM, "GBP")
        contentValues.put(COLUMN_CONVERT_TO, "EUR")
        contentValues.put(COLUMN_MULTIPLIER, "1.11")
        dataBase.insert(TABLE_NAME, null, contentValues)

        // Conversion for GBP to JPY
        contentValues.put(COLUMN_CATEGORY, "Currency")
        contentValues.put(COLUMN_CONVERT_FROM, "GBP")
        contentValues.put(COLUMN_CONVERT_TO, "JPY")
        contentValues.put(COLUMN_MULTIPLIER, "138.48")
        dataBase.insert(TABLE_NAME, null, contentValues)
    }

    /**
     * Conversion data for converting USD into other currencies.
     *
     * @param dataBase the instance of the database that will be used
     * @param contentValues enables the data to be inserted into the database
     */
    private fun conversionInfoUSD(dataBase: SQLiteDatabase, contentValues:
ContentValues) {
        // Conversion for USD to USD
        contentValues.put(COLUMN_CATEGORY, "Currency")
        contentValues.put(COLUMN_CONVERT_FROM, "USD")
        contentValues.put(COLUMN_CONVERT_TO, "USD")
        contentValues.put(COLUMN_MULTIPLIER, "1.00")
        dataBase.insert(TABLE_NAME, null, contentValues)

        // Conversion for USD to GBP
        contentValues.put(COLUMN_CATEGORY, "Currency")
        contentValues.put(COLUMN_CONVERT_FROM, "USD")
        contentValues.put(COLUMN_CONVERT_TO, "GBP")
        contentValues.put(COLUMN_MULTIPLIER, "0.75")
        dataBase.insert(TABLE_NAME, null, contentValues)

        // Conversion for USD to EUR
        contentValues.put(COLUMN_CATEGORY, "Currency")
        contentValues.put(COLUMN_CONVERT_FROM, "USD")
        contentValues.put(COLUMN_CONVERT_TO, "EUR")
        contentValues.put(COLUMN_MULTIPLIER, "0.84")
        dataBase.insert(TABLE_NAME, null, contentValues)

        // Conversion for USD to JPY
        contentValues.put(COLUMN_CATEGORY, "Currency")
        contentValues.put(COLUMN_CONVERT_FROM, "USD")
        contentValues.put(COLUMN_CONVERT_TO, "JPY")
        contentValues.put(COLUMN_MULTIPLIER, "104.07")
        dataBase.insert(TABLE_NAME, null, contentValues)
    }

    /**
     * Conversion data for converting EUR into other currencies.
     *
     * @param dataBase the instance of the database that will be used
     * @param contentValues enables the data to be inserted into the database
     */
    private fun conversionInfoEUR(dataBase: SQLiteDatabase, contentValues:
ContentValues) {
        // Conversion for EUR to EUR
        contentValues.put(COLUMN_CATEGORY, "Currency")
        contentValues.put(COLUMN_CONVERT_FROM, "EUR")
        contentValues.put(COLUMN_CONVERT_TO, "EUR")
        contentValues.put(COLUMN_MULTIPLIER, "1.00")
        dataBase.insert(TABLE_NAME, null, contentValues)

        // Conversion for EUR to GBP
        contentValues.put(COLUMN_CATEGORY, "Currency")
        contentValues.put(COLUMN_CONVERT_FROM, "EUR")
        contentValues.put(COLUMN_CONVERT_TO, "GBP")
        contentValues.put(COLUMN_MULTIPLIER, "0.90")
        dataBase.insert(TABLE_NAME, null, contentValues)

        // Conversion for EUR to USD
        contentValues.put(COLUMN_CATEGORY, "Currency")
        contentValues.put(COLUMN_CONVERT_FROM, "EUR")
        contentValues.put(COLUMN_CONVERT_TO, "USD")
        contentValues.put(COLUMN_MULTIPLIER, "1.20")
        dataBase.insert(TABLE_NAME, null, contentValues)

        // Conversion for EUR to JPY
        contentValues.put(COLUMN_CATEGORY, "Currency")
        contentValues.put(COLUMN_CONVERT_FROM, "EUR")
        contentValues.put(COLUMN_CONVERT_TO, "JPY")
        contentValues.put(COLUMN_MULTIPLIER, "124.50")
        dataBase.insert(TABLE_NAME, null, contentValues)
    }

    /**
     * Conversion data for converting JPY into other currencies.
     *
     * @param dataBase the instance of the database that will be used
     * @param contentValues enables the data to be inserted into the database
     */
    private fun conversionInfoJPY(dataBase: SQLiteDatabase, contentValues:
ContentValues) {
        // Conversion for JPY to JPY
        contentValues.put(COLUMN_CATEGORY, "Currency")
        contentValues.put(COLUMN_CONVERT_FROM, "JPY")
        contentValues.put(COLUMN_CONVERT_TO, "JPY")
        contentValues.put(COLUMN_MULTIPLIER, "1.00")
        dataBase.insert(TABLE_NAME, null, contentValues)

        // Conversion for JPY to GBP
        contentValues.put(COLUMN_CATEGORY, "Currency")
        contentValues.put(COLUMN_CONVERT_FROM, "JPY")
        contentValues.put(COLUMN_CONVERT_TO, "GBP")
        contentValues.put(COLUMN_MULTIPLIER, "0.0072")
        dataBase.insert(TABLE_NAME, null, contentValues)

        // Conversion for JPY to USD
        contentValues.put(COLUMN_CATEGORY, "Currency")
        contentValues.put(COLUMN_CONVERT_FROM, "JPY")
        contentValues.put(COLUMN_CONVERT_TO, "USD")
        contentValues.put(COLUMN_MULTIPLIER, "0.0096")
        dataBase.insert(TABLE_NAME, null, contentValues)

        // Conversion for EUR to JPY
        contentValues.put(COLUMN_CATEGORY, "Currency")
        contentValues.put(COLUMN_CONVERT_FROM, "JPY")
        contentValues.put(COLUMN_CONVERT_TO, "EUR")
        contentValues.put(COLUMN_MULTIPLIER, "0.0080")
        dataBase.insert(TABLE_NAME, null, contentValues)
    }

    /**
     * Conversion data for converting Centimeters into other measurements.
     *
     * @param dataBase the instance of the database that will be used
     * @param contentValues enables the data to be inserted into the database
     */
    private fun conversionInfoCentimeters(dataBase: SQLiteDatabase, contentValues:
ContentValues) {
        // Conversion for Centimeters to Centimeters
        contentValues.put(COLUMN_CATEGORY, "Measurements")
        contentValues.put(COLUMN_CONVERT_FROM, "Centimeters")
        contentValues.put(COLUMN_CONVERT_TO, "Centimeters")
        contentValues.put(COLUMN_MULTIPLIER, "1.00")
        dataBase.insert(TABLE_NAME, null, contentValues)

        // Conversion for Centimeters to Inches
        contentValues.put(COLUMN_CATEGORY, "Measurements")
        contentValues.put(COLUMN_CONVERT_FROM, "Centimeters")
        contentValues.put(COLUMN_CONVERT_TO, "Inches")
        contentValues.put(COLUMN_MULTIPLIER, "0.393701")
        dataBase.insert(TABLE_NAME, null, contentValues)

        // Conversion for Centimeters to Millimeters
        contentValues.put(COLUMN_CATEGORY, "Measurements")
        contentValues.put(COLUMN_CONVERT_FROM, "Centimeters")
        contentValues.put(COLUMN_CONVERT_TO, "Millimeters")
        contentValues.put(COLUMN_MULTIPLIER, "10.00")
        dataBase.insert(TABLE_NAME, null, contentValues)

        // Conversion for Centimeters to Foot
        contentValues.put(COLUMN_CATEGORY, "Measurements")
        contentValues.put(COLUMN_CONVERT_FROM, "Centimeters")
        contentValues.put(COLUMN_CONVERT_TO, "Foot")
        contentValues.put(COLUMN_MULTIPLIER, "0.0328084")
        dataBase.insert(TABLE_NAME, null, contentValues)
    }

    /**
     * Conversion data for converting Inches into other measurements.
     *
     * @param dataBase the instance of the database that will be used
     * @param contentValues enables the data to be inserted into the database
     */
    private fun conversionInfoInches(dataBase: SQLiteDatabase, contentValues:
ContentValues) {
        // Conversion for Inches to Inches
        contentValues.put(COLUMN_CATEGORY, "Measurements")
        contentValues.put(COLUMN_CONVERT_FROM, "Inches")
        contentValues.put(COLUMN_CONVERT_TO, "Inches")
        contentValues.put(COLUMN_MULTIPLIER, "1.00")
        dataBase.insert(TABLE_NAME, null, contentValues)

        // Conversion for Inches to Centimeters
        contentValues.put(COLUMN_CATEGORY, "Measurements")
        contentValues.put(COLUMN_CONVERT_FROM, "Inches")
        contentValues.put(COLUMN_CONVERT_TO, "Centimeters")
        contentValues.put(COLUMN_MULTIPLIER, "2.54")
        dataBase.insert(TABLE_NAME, null, contentValues)

        // Conversion for Inches to Millimeters
        contentValues.put(COLUMN_CATEGORY, "Measurements")
        contentValues.put(COLUMN_CONVERT_FROM, "Inches")
        contentValues.put(COLUMN_CONVERT_TO, "Millimeters")
        contentValues.put(COLUMN_MULTIPLIER, "25.40")
        dataBase.insert(TABLE_NAME, null, contentValues)

        // Conversion for Inches to Foot
        contentValues.put(COLUMN_CATEGORY, "Measurements")
        contentValues.put(COLUMN_CONVERT_FROM, "Inches")
        contentValues.put(COLUMN_CONVERT_TO, "Foot")
        contentValues.put(COLUMN_MULTIPLIER, "0.0833333")
        dataBase.insert(TABLE_NAME, null, contentValues)
    }

    /**
     * Conversion data for converting Inches into other measurements.
     *
     * @param dataBase the instance of the database that will be used
     * @param contentValues enables the data to be inserted into the database
     */
    private fun conversionInfoMillimeters(dataBase: SQLiteDatabase, contentValues:
ContentValues) {
        // Conversion for Millimeters to Millimeters
        contentValues.put(COLUMN_CATEGORY, "Measurements")
        contentValues.put(COLUMN_CONVERT_FROM, "Millimeters")
        contentValues.put(COLUMN_CONVERT_TO, "Millimeters")
        contentValues.put(COLUMN_MULTIPLIER, "1.00")
        dataBase.insert(TABLE_NAME, null, contentValues)

        // Conversion for Millimeters to Centimeters
        contentValues.put(COLUMN_CATEGORY, "Measurements")
        contentValues.put(COLUMN_CONVERT_FROM, "Millimeters")
        contentValues.put(COLUMN_CONVERT_TO, "Centimeters")
        contentValues.put(COLUMN_MULTIPLIER, "0.10")
        dataBase.insert(TABLE_NAME, null, contentValues)

        // Conversion for Millimeters to Inches
        contentValues.put(COLUMN_CATEGORY, "Measurements")
        contentValues.put(COLUMN_CONVERT_FROM, "Millimeters")
        contentValues.put(COLUMN_CONVERT_TO, "Inches")
        contentValues.put(COLUMN_MULTIPLIER, "0.0393701")
        dataBase.insert(TABLE_NAME, null, contentValues)

        // Conversion for Millimeters to Foot
        contentValues.put(COLUMN_CATEGORY, "Measurements")
        contentValues.put(COLUMN_CONVERT_FROM, "Millimeters")
        contentValues.put(COLUMN_CONVERT_TO, "Foot")
        contentValues.put(COLUMN_MULTIPLIER, "0.003280841666667")
        dataBase.insert(TABLE_NAME, null, contentValues)
    }

    /**
     * Conversion data for converting Foot into other measurements.
     *
     * @param dataBase the instance of the database that will be used
     * @param contentValues enables the data to be inserted into the database
     */
    private fun conversionInfoFoot(dataBase: SQLiteDatabase, contentValues:
ContentValues) {
        // Conversion for Foot to Foot
        contentValues.put(COLUMN_CATEGORY, "Measurements")
        contentValues.put(COLUMN_CONVERT_FROM, "Foot")
        contentValues.put(COLUMN_CONVERT_TO, "Foot")
        contentValues.put(COLUMN_MULTIPLIER, "1.00")
        dataBase.insert(TABLE_NAME, null, contentValues)

        // Conversion for Foot to Centimeters
        contentValues.put(COLUMN_CATEGORY, "Measurements")
        contentValues.put(COLUMN_CONVERT_FROM, "Foot")
        contentValues.put(COLUMN_CONVERT_TO, "Centimeters")
        contentValues.put(COLUMN_MULTIPLIER, "30.48")
        dataBase.insert(TABLE_NAME, null, contentValues)

        // Conversion for Foot to Inches
        contentValues.put(COLUMN_CATEGORY, "Measurements")
        contentValues.put(COLUMN_CONVERT_FROM, "Foot")
        contentValues.put(COLUMN_CONVERT_TO, "Inches")
        contentValues.put(COLUMN_MULTIPLIER, "12.00")
        dataBase.insert(TABLE_NAME, null, contentValues)

        // Conversion for Foot to Millimeters
        contentValues.put(COLUMN_CATEGORY, "Measurements")
        contentValues.put(COLUMN_CONVERT_FROM, "Foot")
        contentValues.put(COLUMN_CONVERT_TO, "Millimeters")
        contentValues.put(COLUMN_MULTIPLIER, "304.80")
        dataBase.insert(TABLE_NAME, null, contentValues)
    }

    /**
     * Conversion data for converting Stone into other weights.
     *
     * @param dataBase the instance of the database that will be used
     * @param contentValues enables the data to be inserted into the database
     */
    private fun conversionInfoStone(dataBase: SQLiteDatabase, contentValues:
ContentValues) {
        // Conversion for Stone to Stone
        contentValues.put(COLUMN_CATEGORY, "Weights")
        contentValues.put(COLUMN_CONVERT_FROM, "Stone")
        contentValues.put(COLUMN_CONVERT_TO, "Stone")
        contentValues.put(COLUMN_MULTIPLIER, "1.00")
        dataBase.insert(TABLE_NAME, null, contentValues)

        // Conversion for Stone to Pounds
        contentValues.put(COLUMN_CATEGORY, "Weights")
        contentValues.put(COLUMN_CONVERT_FROM, "Stone")
        contentValues.put(COLUMN_CONVERT_TO, "Pounds")
        contentValues.put(COLUMN_MULTIPLIER, "14.00")
        dataBase.insert(TABLE_NAME, null, contentValues)

        // Conversion for Stone to Grams
        contentValues.put(COLUMN_CATEGORY, "Weights")
        contentValues.put(COLUMN_CONVERT_FROM, "Stone")
        contentValues.put(COLUMN_CONVERT_TO, "Grams")
        contentValues.put(COLUMN_MULTIPLIER, "6350.29")
        dataBase.insert(TABLE_NAME, null, contentValues)

        // Conversion for Stone to Kilograms
        contentValues.put(COLUMN_CATEGORY, "Weights")
        contentValues.put(COLUMN_CONVERT_FROM, "Stone")
        contentValues.put(COLUMN_CONVERT_TO, "Kilograms")
        contentValues.put(COLUMN_MULTIPLIER, "6.35029")
        dataBase.insert(TABLE_NAME, null, contentValues)
    }

    /**
     * Conversion data for converting Pounds into other weights.
     *
     * @param dataBase the instance of the database that will be used
     * @param contentValues enables the data to be inserted into the database
     */
    private fun conversionInfoPounds(dataBase: SQLiteDatabase, contentValues:
ContentValues) {
        // Conversion for Pounds to Pounds
        contentValues.put(COLUMN_CATEGORY, "Weights")
        contentValues.put(COLUMN_CONVERT_FROM, "Pounds")
        contentValues.put(COLUMN_CONVERT_TO, "Pounds")
        contentValues.put(COLUMN_MULTIPLIER, "1.00")
        dataBase.insert(TABLE_NAME, null, contentValues)

        // Conversion for Pounds to Stone
        contentValues.put(COLUMN_CATEGORY, "Weights")
        contentValues.put(COLUMN_CONVERT_FROM, "Pounds")
        contentValues.put(COLUMN_CONVERT_TO, "Stone")
        contentValues.put(COLUMN_MULTIPLIER, "0.0714286")
        dataBase.insert(TABLE_NAME, null, contentValues)

        // Conversion for Pounds to Grams
        contentValues.put(COLUMN_CATEGORY, "Weights")
        contentValues.put(COLUMN_CONVERT_FROM, "Pounds")
        contentValues.put(COLUMN_CONVERT_TO, "Grams")
        contentValues.put(COLUMN_MULTIPLIER, "453.592")
        dataBase.insert(TABLE_NAME, null, contentValues)

        // Conversion for Pounds to Kilograms
        contentValues.put(COLUMN_CATEGORY, "Weights")
        contentValues.put(COLUMN_CONVERT_FROM, "Pounds")
        contentValues.put(COLUMN_CONVERT_TO, "Kilograms")
        contentValues.put(COLUMN_MULTIPLIER, "0.453592")
        dataBase.insert(TABLE_NAME, null, contentValues)
    }

    /**
     * Conversion data for converting Grams into other weights.
     *
     * @param dataBase the instance of the database that will be used
     * @param contentValues enables the data to be inserted into the database
     */
    private fun conversionInfoGrams(dataBase: SQLiteDatabase, contentValues:
ContentValues) {
        // Conversion for Grams to Grams
        contentValues.put(COLUMN_CATEGORY, "Weights")
        contentValues.put(COLUMN_CONVERT_FROM, "Grams")
        contentValues.put(COLUMN_CONVERT_TO, "Grams")
        contentValues.put(COLUMN_MULTIPLIER, "1.00")
        dataBase.insert(TABLE_NAME, null, contentValues)

        // Conversion for Grams to Stone
        contentValues.put(COLUMN_CATEGORY, "Weights")
        contentValues.put(COLUMN_CONVERT_FROM, "Grams")
        contentValues.put(COLUMN_CONVERT_TO, "Stone")
        contentValues.put(COLUMN_MULTIPLIER, "0.000157473")
        dataBase.insert(TABLE_NAME, null, contentValues)

        // Conversion for Grams to Pounds
        contentValues.put(COLUMN_CATEGORY, "Weights")
        contentValues.put(COLUMN_CONVERT_FROM, "Grams")
        contentValues.put(COLUMN_CONVERT_TO, "Pounds")
        contentValues.put(COLUMN_MULTIPLIER, "0.00220462")
        dataBase.insert(TABLE_NAME, null, contentValues)

        // Conversion for Grams to Kilograms
        contentValues.put(COLUMN_CATEGORY, "Weights")
        contentValues.put(COLUMN_CONVERT_FROM, "Grams")
        contentValues.put(COLUMN_CONVERT_TO, "Kilograms")
        contentValues.put(COLUMN_MULTIPLIER, "0.001")
        dataBase.insert(TABLE_NAME, null, contentValues)
    }

    /**
     * Conversion data for converting Kilograms into other weights.
     *
     * @param dataBase the instance of the database that will be used
     * @param contentValues enables the data to be inserted into the database
     */
    private fun conversionInfoKilograms(dataBase: SQLiteDatabase, contentValues:
ContentValues) {
        // Conversion for KiloGrams to KiloGrams
        contentValues.put(COLUMN_CATEGORY, "Weights")
        contentValues.put(COLUMN_CONVERT_FROM, "KiloGrams")
        contentValues.put(COLUMN_CONVERT_TO, "KiloGrams")
        contentValues.put(COLUMN_MULTIPLIER, "1.00")
        dataBase.insert(TABLE_NAME, null, contentValues)

        // Conversion for KiloGrams to Stone
        contentValues.put(COLUMN_CATEGORY, "Weights")
        contentValues.put(COLUMN_CONVERT_FROM, "KiloGrams")
        contentValues.put(COLUMN_CONVERT_TO, "Stone")
        contentValues.put(COLUMN_MULTIPLIER, "0.157473")
        dataBase.insert(TABLE_NAME, null, contentValues)

        // Conversion for KiloGrams to Pounds
        contentValues.put(COLUMN_CATEGORY, "Weights")
        contentValues.put(COLUMN_CONVERT_FROM, "KiloGrams")
        contentValues.put(COLUMN_CONVERT_TO, "Pounds")
        contentValues.put(COLUMN_MULTIPLIER, "2.20462")
        dataBase.insert(TABLE_NAME, null, contentValues)

        // Conversion for KiloGrams to Grams
        contentValues.put(COLUMN_CATEGORY, "Weights")
        contentValues.put(COLUMN_CONVERT_FROM, "KiloGrams")
        contentValues.put(COLUMN_CONVERT_TO, "Grams")
        contentValues.put(COLUMN_MULTIPLIER, "1000.00")
        dataBase.insert(TABLE_NAME, null, contentValues)
    }

    /**
     * Conversion data for converting Liters into other volumes.
     *
     * @param dataBase the instance of the database that will be used
     * @param contentValues enables the data to be inserted into the database
     */
    private fun conversionInfoLiters(dataBase: SQLiteDatabase, contentValues:
ContentValues) {
        // Conversion for Liters to Liters
        contentValues.put(COLUMN_CATEGORY, "Volumes")
        contentValues.put(COLUMN_CONVERT_FROM, "Liters")
        contentValues.put(COLUMN_CONVERT_TO, "Liters")
        contentValues.put(COLUMN_MULTIPLIER, "1.00")
        dataBase.insert(TABLE_NAME, null, contentValues)

        // Conversion for Liters to Millilitres
        contentValues.put(COLUMN_CATEGORY, "Volumes")
        contentValues.put(COLUMN_CONVERT_FROM, "Liters")
        contentValues.put(COLUMN_CONVERT_TO, "Millilitres")
        contentValues.put(COLUMN_MULTIPLIER, "1000")
        dataBase.insert(TABLE_NAME, null, contentValues)

        // Conversion for Liters to Pints
        contentValues.put(COLUMN_CATEGORY, "Volumes")
        contentValues.put(COLUMN_CONVERT_FROM, "Liters")
        contentValues.put(COLUMN_CONVERT_TO, "Pints")
        contentValues.put(COLUMN_MULTIPLIER, "1.75975")
        dataBase.insert(TABLE_NAME, null, contentValues)

        // Conversion for Liters to Gallons
        contentValues.put(COLUMN_CATEGORY, "Volumes")
        contentValues.put(COLUMN_CONVERT_FROM, "Liters")
        contentValues.put(COLUMN_CONVERT_TO, "Gallons")
        contentValues.put(COLUMN_MULTIPLIER, "0.219969")
        dataBase.insert(TABLE_NAME, null, contentValues)
    }

    /**
     * Conversion data for converting Liters into other volumes.
     *
     * @param dataBase the instance of the database that will be used
     * @param contentValues enables the data to be inserted into the database
     */
    private fun conversionInfoMillilitres(dataBase: SQLiteDatabase, contentValues:
ContentValues) {
        // Conversion for Millilitres to Millilitres
        contentValues.put(COLUMN_CATEGORY, "Volumes")
        contentValues.put(COLUMN_CONVERT_FROM, "Millilitres")
        contentValues.put(COLUMN_CONVERT_TO, "Millilitres")
        contentValues.put(COLUMN_MULTIPLIER, "1.00")
        dataBase.insert(TABLE_NAME, null, contentValues)

        // Conversion for Millilitres to Liters
        contentValues.put(COLUMN_CATEGORY, "Volumes")
        contentValues.put(COLUMN_CONVERT_FROM, "Millilitres")
        contentValues.put(COLUMN_CONVERT_TO, "Liters")
        contentValues.put(COLUMN_MULTIPLIER, "0.001")
        dataBase.insert(TABLE_NAME, null, contentValues)

        // Conversion for Millilitres to Pints
        contentValues.put(COLUMN_CATEGORY, "Volumes")
        contentValues.put(COLUMN_CONVERT_FROM, "Millilitres")
        contentValues.put(COLUMN_CONVERT_TO, "Pints")
        contentValues.put(COLUMN_MULTIPLIER, "0.00211338")
        dataBase.insert(TABLE_NAME, null, contentValues)

        // Conversion for Millilitres to Gallons
        contentValues.put(COLUMN_CATEGORY, "Volumes")
        contentValues.put(COLUMN_CONVERT_FROM, "Millilitres")
        contentValues.put(COLUMN_CONVERT_TO, "Gallons")
        contentValues.put(COLUMN_MULTIPLIER, "0.000219969")
        dataBase.insert(TABLE_NAME, null, contentValues)
    }

    /**
     * Conversion data for converting Pints into other volumes.
     *
     * @param dataBase the instance of the database that will be used
     * @param contentValues enables the data to be inserted into the database
     */
    private fun conversionInfoPints(dataBase: SQLiteDatabase, contentValues:
ContentValues) {
        // Conversion for Pints to Pints
        contentValues.put(COLUMN_CATEGORY, "Volumes")
        contentValues.put(COLUMN_CONVERT_FROM, "Pints")
        contentValues.put(COLUMN_CONVERT_TO, "Pints")
        contentValues.put(COLUMN_MULTIPLIER, "1.00")
        dataBase.insert(TABLE_NAME, null, contentValues)

        // Conversion for Pints to Liters
        contentValues.put(COLUMN_CATEGORY, "Volumes")
        contentValues.put(COLUMN_CONVERT_FROM, "Pints")
        contentValues.put(COLUMN_CONVERT_TO, "Liters")
        contentValues.put(COLUMN_MULTIPLIER, "0.568261")
        dataBase.insert(TABLE_NAME, null, contentValues)

        // Conversion for Pints to Millilitres
        contentValues.put(COLUMN_CATEGORY, "Volumes")
        contentValues.put(COLUMN_CONVERT_FROM, "Pints")
        contentValues.put(COLUMN_CONVERT_TO, "Millilitres")
        contentValues.put(COLUMN_MULTIPLIER, "568.261")
        dataBase.insert(TABLE_NAME, null, contentValues)

        // Conversion for Pints to Gallons
        contentValues.put(COLUMN_CATEGORY, "Volumes")
        contentValues.put(COLUMN_CONVERT_FROM, "Pints")
        contentValues.put(COLUMN_CONVERT_TO, "Gallons")
        contentValues.put(COLUMN_MULTIPLIER, "0.125")
        dataBase.insert(TABLE_NAME, null, contentValues)
    }

    /**
     * Conversion data for converting Gallons into other volumes.
     *
     * @param dataBase the instance of the database that will be used
     * @param contentValues enables the data to be inserted into the database
     */
    private fun conversionInfoGallons(dataBase: SQLiteDatabase, contentValues:
ContentValues) {
        // Conversion for Pints to Pints
        contentValues.put(COLUMN_CATEGORY, "Volumes")
        contentValues.put(COLUMN_CONVERT_FROM, "Gallons")
        contentValues.put(COLUMN_CONVERT_TO, "Gallons")
        contentValues.put(COLUMN_MULTIPLIER, "1.00")
        dataBase.insert(TABLE_NAME, null, contentValues)

        // Conversion for Gallons to Liters
        contentValues.put(COLUMN_CATEGORY, "Volumes")
        contentValues.put(COLUMN_CONVERT_FROM, "Gallons")
        contentValues.put(COLUMN_CONVERT_TO, "Liters")
        contentValues.put(COLUMN_MULTIPLIER, "4.54609")
        dataBase.insert(TABLE_NAME, null, contentValues)

        // Conversion for Gallons to Millilitres
        contentValues.put(COLUMN_CATEGORY, "Volumes")
        contentValues.put(COLUMN_CONVERT_FROM, "Gallons")
        contentValues.put(COLUMN_CONVERT_TO, "Millilitres")
        contentValues.put(COLUMN_MULTIPLIER, "4546.09")
        dataBase.insert(TABLE_NAME, null, contentValues)

        // Conversion for Gallons to Pints
        contentValues.put(COLUMN_CATEGORY, "Volumes")
        contentValues.put(COLUMN_CONVERT_FROM, "Gallons")
        contentValues.put(COLUMN_CONVERT_TO, "Pints")
        contentValues.put(COLUMN_MULTIPLIER, "8")
        dataBase.insert(TABLE_NAME, null, contentValues)
    }

    /**
     * Conversion data for converting MPH into other volumes.
     *
     * @param dataBase the instance of the database that will be used
     * @param contentValues enables the data to be inserted into the database
     */
    private fun conversionInfoMPH(dataBase: SQLiteDatabase, contentValues:
ContentValues) {
        // Conversion for MPH to MPH
        contentValues.put(COLUMN_CATEGORY, "Speed")
        contentValues.put(COLUMN_CONVERT_FROM, "MPH")
        contentValues.put(COLUMN_CONVERT_TO, "MPH")
        contentValues.put(COLUMN_MULTIPLIER, "1.00")
        dataBase.insert(TABLE_NAME, null, contentValues)

        // Conversion for MPH to KPH
        contentValues.put(COLUMN_CATEGORY, "Speed")
        contentValues.put(COLUMN_CONVERT_FROM, "MPH")
        contentValues.put(COLUMN_CONVERT_TO, "KPH")
        contentValues.put(COLUMN_MULTIPLIER, "3.6")
        dataBase.insert(TABLE_NAME, null, contentValues)

        // Conversion for MPH to Knot
        contentValues.put(COLUMN_CATEGORY, "Speed")
        contentValues.put(COLUMN_CONVERT_FROM, "MPH")
        contentValues.put(COLUMN_CONVERT_TO, "Knot")
        contentValues.put(COLUMN_MULTIPLIER, "1.94384")
        dataBase.insert(TABLE_NAME, null, contentValues)

        // Conversion for MPH to MPS
        contentValues.put(COLUMN_CATEGORY, "Speed")
        contentValues.put(COLUMN_CONVERT_FROM, "MPH")
        contentValues.put(COLUMN_CONVERT_TO, "MPS")
        contentValues.put(COLUMN_MULTIPLIER, "0.51444325460445")
        dataBase.insert(TABLE_NAME, null, contentValues)
    }

    /**
     * Conversion data for converting KPH into other volumes.
     *
     * @param dataBase the instance of the database that will be used
     * @param contentValues enables the data to be inserted into the database
     */
    private fun conversionInfoKPH(dataBase: SQLiteDatabase, contentValues:
ContentValues) {
        // Conversion for KPH to KPH
        contentValues.put(COLUMN_CATEGORY, "Speed")
        contentValues.put(COLUMN_CONVERT_FROM, "KPH")
        contentValues.put(COLUMN_CONVERT_TO, "KPH")
        contentValues.put(COLUMN_MULTIPLIER, "1.00")
        dataBase.insert(TABLE_NAME, null, contentValues)

        // Conversion for KPH to MPH
        contentValues.put(COLUMN_CATEGORY, "Speed")
        contentValues.put(COLUMN_CONVERT_FROM, "KPH")
        contentValues.put(COLUMN_CONVERT_TO, "MPH")
        contentValues.put(COLUMN_MULTIPLIER, "0.621371")
        dataBase.insert(TABLE_NAME, null, contentValues)

        // Conversion for KPH to Knot
        contentValues.put(COLUMN_CATEGORY, "Speed")
        contentValues.put(COLUMN_CONVERT_FROM, "KPH")
        contentValues.put(COLUMN_CONVERT_TO, "Knot")
        contentValues.put(COLUMN_MULTIPLIER, "0.539957")
        dataBase.insert(TABLE_NAME, null, contentValues)

        // Conversion for KPH to MPS
        contentValues.put(COLUMN_CATEGORY, "Speed")
        contentValues.put(COLUMN_CONVERT_FROM, "KPH")
        contentValues.put(COLUMN_CONVERT_TO, "MPS")
        contentValues.put(COLUMN_MULTIPLIER, "0.277778")
        dataBase.insert(TABLE_NAME, null, contentValues)
    }

    /**
     * Conversion data for converting Knot into other volumes.
     *
     * @param dataBase the instance of the database that will be used
     * @param contentValues enables the data to be inserted into the database
     */
    private fun conversionInfoKnot(dataBase: SQLiteDatabase, contentValues:
ContentValues) {
        // Conversion for Knot to Knot
        contentValues.put(COLUMN_CATEGORY, "Speed")
        contentValues.put(COLUMN_CONVERT_FROM, "Knot")
        contentValues.put(COLUMN_CONVERT_TO, "Knot")
        contentValues.put(COLUMN_MULTIPLIER, "1.00")
        dataBase.insert(TABLE_NAME, null, contentValues)

        // Conversion for Knot to MPH
        contentValues.put(COLUMN_CATEGORY, "Speed")
        contentValues.put(COLUMN_CONVERT_FROM, "Knot")
        contentValues.put(COLUMN_CONVERT_TO, "MPH")
        contentValues.put(COLUMN_MULTIPLIER, "1.15078")
        dataBase.insert(TABLE_NAME, null, contentValues)

        // Conversion for Knot to KPH
        contentValues.put(COLUMN_CATEGORY, "Speed")
        contentValues.put(COLUMN_CONVERT_FROM, "Knot")
        contentValues.put(COLUMN_CONVERT_TO, "KPH")
        contentValues.put(COLUMN_MULTIPLIER, "1.852")
        dataBase.insert(TABLE_NAME, null, contentValues)

        // Conversion for Knot to MPS
        contentValues.put(COLUMN_CATEGORY, "Speed")
        contentValues.put(COLUMN_CONVERT_FROM, "Knot")
        contentValues.put(COLUMN_CONVERT_TO, "MPS")
        contentValues.put(COLUMN_MULTIPLIER, "0.514444")
        dataBase.insert(TABLE_NAME, null, contentValues)
    }

    /**
     * Conversion data for converting MPS into other volumes.
     *
     * @param dataBase the instance of the database that will be used
     * @param contentValues enables the data to be inserted into the database
     */
    private fun conversionInfoMPS(dataBase: SQLiteDatabase, contentValues:
ContentValues) {
        // Conversion for MPS to MPS
        contentValues.put(COLUMN_CATEGORY, "Speed")
        contentValues.put(COLUMN_CONVERT_FROM, "MPS")
        contentValues.put(COLUMN_CONVERT_TO, "Knot")
        contentValues.put(COLUMN_MULTIPLIER, "1.00")
        dataBase.insert(TABLE_NAME, null, contentValues)

        // Conversion for MPS to MPH
        contentValues.put(COLUMN_CATEGORY, "Speed")
        contentValues.put(COLUMN_CONVERT_FROM, "MPS")
        contentValues.put(COLUMN_CONVERT_TO, "MPH")
        contentValues.put(COLUMN_MULTIPLIER, "2.23694")
        dataBase.insert(TABLE_NAME, null, contentValues)

        // Conversion for MPS to KPH
        contentValues.put(COLUMN_CATEGORY, "Speed")
        contentValues.put(COLUMN_CONVERT_FROM, "MPS")
        contentValues.put(COLUMN_CONVERT_TO, "KPH")
        contentValues.put(COLUMN_MULTIPLIER, "3.6")
        dataBase.insert(TABLE_NAME, null, contentValues)

        // Conversion for MPS to Knot
        contentValues.put(COLUMN_CATEGORY, "Speed")
        contentValues.put(COLUMN_CONVERT_FROM, "MPS")
        contentValues.put(COLUMN_CONVERT_TO, "Knot")
        contentValues.put(COLUMN_MULTIPLIER, "1.94384")
        dataBase.insert(TABLE_NAME, null, contentValues)
    }
}
```

6.9. Activity Main Xml File

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">

        <androidx.appcompat.widget.Toolbar
            android:id="@+id/toolbar"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:background="@color/colorPrimary"
            android:elevation="6dp"
            app:titleTextColor="#FFFFFF" />

        <com.google.android.material.tabs.TabLayout
            android:id="@+id/tab_layout"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:background="@color/colorPrimary"
            android:elevation="6dp"
            android:minHeight="?attr/actionBarSize"
            android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar"/>

        <androidx.viewpager.widget.ViewPager
            android:id="@+id/pager"
            android:layout_width="match_parent"
            android:layout_height="fill_parent"
            android:background="#000000" />
    </LinearLayout>

</androidx.constraintlayout.widget.ConstraintLayout>
```

6.10. Page One Fragment Xml File

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#000000">

    <TextView
        android:id="@+id/unitSelectorTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/UnitSelectorLabel"
        android:textColor="@android:color/white"
        android:textSize="20sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.053"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.065" />

    <Spinner
        android:id="@+id/unitSpinner"
        android:layout_width="377dp"
        android:layout_height="32dp"
        android:background="@android:color/white"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.47"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.125" />

    <TextView
        android:id="@+id/valueTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/ValueLabel"
        android:textColor="@android:color/white"
        android:textSize="20sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.044"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.333" />

    <Spinner
        android:id="@+id/valueSpinner"
        android:layout_width="377dp"
        android:layout_height="32dp"
        android:background="@android:color/white"
        android:scrollbarSize="20dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.444"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.396" />

    <TextView
        android:id="@+id/conversionTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/conversionLabel"
        android:textColor="@android:color/white"
        android:textSize="20sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.051"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.6" />

    <Spinner
        android:id="@+id/conversionSpinner"
        android:layout_width="377dp"
        android:layout_height="32dp"
        android:background="@android:color/white"
        android:scrollbarSize="20dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.47"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.663" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

6.11. Page One Fragment Landscape Xml File

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#000000">

    <TextView
        android:id="@+id/unitSelectorTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/UnitSelectorLabel"
        android:textColor="@android:color/white"
        android:textSize="20sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.053"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.065" />

    <Spinner
        android:id="@+id/unitSpinner"
        android:layout_width="377dp"
        android:layout_height="32dp"
        android:background="@android:color/white"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.093"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.187" />

    <Spinner
        android:id="@+id/valueSpinner"
        android:layout_width="377dp"
        android:layout_height="32dp"
        android:background="@android:color/white"
        android:scrollbarSize="20dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.093"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.53" />

    <TextView
        android:id="@+id/valueTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/ValueLabel"
        android:textColor="@android:color/white"
        android:textSize="20sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.048"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.414" />

    <Spinner
        android:id="@+id/conversionSpinner"
        android:layout_width="377dp"
        android:layout_height="32dp"
        android:background="@android:color/white"
        android:scrollbarSize="20dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.093"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.865" />

    <TextView
        android:id="@+id/conversionTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/conversionLabel"
        android:textColor="@android:color/white"
        android:textSize="20sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.052"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.747" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

6.12. Page Two Fragment Xml File

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#000000">

    <TextView
        android:id="@+id/valueSelectedTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/valueSelectedLabel"
        android:textColor="@android:color/white"
        android:textSize="20sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.054"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.068" />

    <TextView
        android:id="@+id/selectedValueDisplayTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:hint="@string/valueHint"
        android:text="@string/selectedValueDisplayLabel"
        android:textColor="@android:color/white"
        android:textSize="15sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.049"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.13" />

    <TextView
        android:id="@+id/conversionSelectedTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/conversionSelectedLabel"
        android:textColor="@android:color/white"
        android:textSize="20sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.057"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.247" />

    <TextView
        android:id="@+id/selectedConversionDisplayTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:hint="@string/conversionHint"
        android:text="@string/selectedConversionDisplayLabel"
        android:textColor="@android:color/white"
        android:textSize="15sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.047"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.308" />

    <TableLayout
        android:id="@+id/tableLayout"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="10dp"
        android:stretchColumns="*"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent">

        <TableRow
            android:layout_width="match_parent"
            android:layout_height="match_parent">

            <EditText
                android:id="@+id/calculationInputTextView"
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:layout_span="3"
                android:autofillHints=""
                android:clickable="false"
                android:ems="10"
                android:focusable="false"
                android:focusableInTouchMode="false"
                android:hint="@string/calculationInputLabel"
                android:inputType="number"
                android:textColor="@android:color/white"
                android:textColorHint="@android:color/darker_gray" />
        </TableRow>

        <TableRow
            android:layout_width="match_parent"
            android:layout_height="match_parent">

            <EditText
                android:id="@+id/calculationResultTextView"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_span="3"
                android:autofillHints=""
                android:clickable="false"
                android:ems="10"
                android:focusable="false"
                android:focusableInTouchMode="false"
                android:hint="@string/calculationResultLabel"
                android:inputType="number"
                android:textColor="@android:color/white"
                android:textColorHint="@android:color/darker_gray" />

        </TableRow>
        <TableRow
            android:layout_width="match_parent"
            android:layout_height="match_parent">

            <Button
                android:id="@+id/calculatorButtonNumber7"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="@string/calculatorButtonNumber7" />

            <Button
                android:id="@+id/calculatorButtonNumber8"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="@string/calculatorButtonNumber8" />

            <Button
                android:id="@+id/calculatorButtonNumber9"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="@string/calculatorButtonNumber9" />
        </TableRow>

        <TableRow
            android:layout_width="match_parent"
            android:layout_height="match_parent">

            <Button
                android:id="@+id/calculatorButtonNumber4"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="@string/calculatorButtonNumber4" />

            <Button
                android:id="@+id/calculatorButtonNumber5"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="@string/calculatorButtonNumber5" />

            <Button
                android:id="@+id/calculatorButtonNumber6"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="@string/calculatorButtonNumber6" />
        </TableRow>

        <TableRow
            android:layout_width="match_parent"
            android:layout_height="match_parent">

            <Button
                android:id="@+id/calculatorButtonNumber1"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="@string/calculatorButtonNumber1" />

            <Button
                android:id="@+id/calculatorButtonNumber2"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="@string/calculatorButtonNumber2" />

            <Button
                android:id="@+id/calculatorButtonNumber3"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="@string/calculatorButtonNumber3" />
        </TableRow>

        <TableRow
            android:layout_width="match_parent"
            android:layout_height="match_parent">

            <Button
                android:id="@+id/calculatorButtonClear"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="@string/calculatorButtonClear" />

            <Button
                android:id="@+id/calculatorButtonNumber0"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="@string/calculatorButton0" />

            <Button
                android:id="@+id/calculatorButtonDot"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="@string/calculatorButtonDot" />

        </TableRow>

        <TableRow
            android:layout_width="match_parent"
            android:layout_height="match_parent" >

            <Button
                android:id="@+id/calculatorButtonEnter"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_span="3"
                android:text="@string/calculatorButtonConvert" />
        </TableRow>

    </TableLayout>

</androidx.constraintlayout.widget.ConstraintLayout>
```

6.13. Page Two Fragment Landscape Xml File

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#000000">

    <TableLayout
        android:id="@+id/tableLayout"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:stretchColumns="*"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent">

        <TableRow
            android:layout_width="match_parent"
            android:layout_height="match_parent">

            <EditText
                android:id="@+id/calculationInputTextView"
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:layout_span="3"
                android:autofillHints=""
                android:clickable="false"
                android:ems="10"
                android:focusable="false"
                android:focusableInTouchMode="false"
                android:hint="@string/calculationInputLabel"
                android:inputType="number"
                android:textColor="@android:color/white"
                android:textColorHint="@android:color/darker_gray" />

            <TextView
                android:id="@+id/valueSelectedTextView"
                android:layout_width="214dp"
                android:layout_height="40dp"
                android:text="@string/valueSelectedLabel"
                android:textColor="@android:color/white"
                android:textSize="20sp" />
        </TableRow>

        <TableRow
            android:layout_width="match_parent"
            android:layout_height="match_parent">

            <EditText
                android:id="@+id/calculationResultTextView"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_span="3"
                android:autofillHints=""
                android:clickable="false"
                android:ems="10"
                android:focusable="false"
                android:focusableInTouchMode="false"
                android:hint="@string/calculationResultLabel"
                android:inputType="number"
                android:textColor="@android:color/white"
                android:textColorHint="@android:color/darker_gray" />

            <TextView
                android:id="@+id/selectedValueDisplayTextView"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:hint="@string/valueHint"
                android:text="@string/selectedValueDisplayLabel"
                android:textColor="@android:color/white"
                android:textSize="15sp" />

        </TableRow>

        <TableRow
            android:layout_width="match_parent"
            android:layout_height="match_parent">

            <Button
                android:id="@+id/calculatorButtonNumber7"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="@string/calculatorButtonNumber7" />

            <Button
                android:id="@+id/calculatorButtonNumber8"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="@string/calculatorButtonNumber8" />

            <Button
                android:id="@+id/calculatorButtonNumber9"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="@string/calculatorButtonNumber9" />

            <TextView
                android:id="@+id/conversionSelectedTextView"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="@string/conversionSelectedLabel"
                android:textColor="@android:color/white"
                android:textSize="20sp" />
        </TableRow>

        <TableRow
            android:layout_width="match_parent"
            android:layout_height="match_parent">

            <Button
                android:id="@+id/calculatorButtonNumber4"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="@string/calculatorButtonNumber4" />
            <Button
                android:id="@+id/calculatorButtonNumber5"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="@string/calculatorButtonNumber5" />

            <Button
                android:id="@+id/calculatorButtonNumber6"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="@string/calculatorButtonNumber6" />

            <TextView
                android:id="@+id/selectedConversionDisplayTextView"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:hint="@string/conversionHint"
                android:text="@string/selectedConversionDisplayLabel"
                android:textColor="@android:color/white"
                android:textSize="15sp" />
        </TableRow>

        <TableRow
            android:layout_width="match_parent"
            android:layout_height="match_parent">

            <Button
                android:id="@+id/calculatorButtonNumber1"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="@string/calculatorButtonNumber1" />

            <Button
                android:id="@+id/calculatorButtonNumber2"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="@string/calculatorButtonNumber2" />

            <Button
                android:id="@+id/calculatorButtonNumber3"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="@string/calculatorButtonNumber3" />
        </TableRow>

        <TableRow
            android:layout_width="match_parent"
            android:layout_height="match_parent">

            <Button
                android:id="@+id/calculatorButtonClear"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="@string/calculatorButtonClear" />

            <Button
                android:id="@+id/calculatorButtonNumber0"
                android:layout_width="wrap_content"
```

```
                android:layout_height="wrap_content"
                android:text="@string/calculatorButton0" />

            <Button
                android:id="@+id/calculatorButtonDot"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="@string/calculatorButtonDot" />

            <Button
                android:id="@+id/calculatorButtonEnter"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_span="3"
                android:text="@string/calculatorButtonConvert" />

        </TableRow>

        <TableRow
            android:layout_width="match_parent"
            android:layout_height="match_parent" >

        </TableRow>

    </TableLayout>

</androidx.constraintlayout.widget.ConstraintLayout>
```

6.14. Page Three Fragment Xml File

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@android:color/black">

    <TableLayout
        android:id="@+id/tableLayout"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:stretchColumns="*"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">

        <TableRow
            android:layout_width="match_parent"
            android:layout_height="match_parent"

            android:paddingBottom="10dp">

            <EditText
                android:id="@+id/newUnitInputEditText"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_span="3"
                android:ems="10"
                android:hint="@string/newUnitInputLabelHint"
                android:inputType="textPersonName"
                android:textColor="@android:color/white"
                android:textColorHint="@android:color/darker_gray" />
        </TableRow>

        <TableRow
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:paddingTop="10dp"
            android:paddingBottom="10dp">

            <EditText
                android:id="@+id/newValueInputEditText"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:ems="10"
                android:hint="@string/newValueInputHint"
                android:inputType="textPersonName"
                android:textColor="@android:color/white"
                android:textColorHint="@android:color/darker_gray"
                android:layout_span="3" />
        </TableRow>

        <TableRow
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:paddingTop="10dp"
            android:paddingBottom="10dp">
            <EditText
                android:id="@+id/newMultiplierInputEditText"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_span="3"
                android:ems="10"
                android:hint="@string/newMultiplierInputHint"
                android:inputType="textPersonName"
                android:textColor="@android:color/white"
                android:textColorHint="@android:color/darker_gray" />

        </TableRow>

        <TableRow
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:paddingTop="10dp"
            android:paddingBottom="10dp">

            <EditText
                android:id="@+id/newConversionInputEditText"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_span="3"
                android:ems="10"
                android:hint="@string/newConversionInputValueHint"
                android:inputType="textPersonName"
                android:textColor="@android:color/white"
                android:textColorHint="@android:color/darker_gray" />
        </TableRow>

        <TableRow
            android:layout_width="match_parent"
            android:layout_height="match_parent">

            <Button
                android:id="@+id/insertButton"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="@string/insertButton" />

            <Button
                android:id="@+id/displayButton"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="@string/displayButton" />

        </TableRow>

        <TableRow
            android:layout_width="match_parent"
            android:layout_height="match_parent">

            <EditText
                android:id="@+id/deleteDatabaseValueNumberInput"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:ems="10"
```

```
                android:hint="@string/deleteDatabaseValueHint"
                android:inputType="number"
                android:layout_span="2"
                android:textColor="@android:color/white"
                android:textColorHint="@android:color/darker_gray" />

            <Button
                android:id="@+id/deleteButton"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="@string/deleteButton" />
        </TableRow>

        <TableRow
            android:layout_width="match_parent"
            android:layout_height="match_parent" >

            <ScrollView
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:layout_span="3">

                <TextView
                    android:id="@+id/dataDisplayTextView"
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:textColor="@android:color/white" />
            </ScrollView>

        </TableRow>

    </TableLayout>

</androidx.constraintlayout.widget.ConstraintLayout>
```

6.14. Page Three Fragment Landscape Xml File

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@android:color/black">

    <TableLayout
        android:id="@+id/tableLayout"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:stretchColumns="*"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">

        <TableRow
            android:layout_width="match_parent"
            android:layout_height="match_parent"

            android:paddingBottom="10dp">

            <EditText
                android:id="@+id/newUnitInputEditText"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_span="4"
                android:ems="10"
                android:hint="@string/newUnitInputLabelHint"
                android:inputType="textPersonName"
                android:textColor="@android:color/white"
                android:textColorHint="@android:color/darker_gray" />

            <Button
                android:id="@+id/insertButton"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="@string/insertButton" />
        </TableRow>

        <TableRow
            android:layout_width="match_parent"
            android:layout_height="match_parent">

            <EditText
                android:id="@+id/newValueInputEditText"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:ems="10"
                android:hint="@string/newValueInputHint"
                android:inputType="textPersonName"
                android:textColor="@android:color/white"
                android:textColorHint="@android:color/darker_gray"
                android:layout_span="4" />

            <Button
                android:id="@+id/displayButton"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="@string/displayButton" />
        </TableRow>

        <TableRow
            android:layout_width="match_parent"
            android:layout_height="match_parent">

            <EditText
                android:id="@+id/newMultiplierInputEditText"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_span="4"
                android:ems="10"
                android:hint="@string/newMultiplierInputHint"
                android:inputType="textPersonName"
                android:textColor="@android:color/white"
                android:textColorHint="@android:color/darker_gray" />

            <EditText
                android:id="@+id/deleteDatabaseValueNumberInput"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_span="1"
                android:ems="10"
                android:hint="@string/deleteDatabaseValueHint"
                android:inputType="number"
                android:textColor="@android:color/white"
                android:textColorHint="@android:color/darker_gray" />

        </TableRow>

        <TableRow
            android:layout_width="match_parent"
            android:layout_height="match_parent">

            <EditText
                android:id="@+id/newConversionInputEditText"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_span="4"
                android:ems="10"
                android:hint="@string/newConversionInputValueHint"
                android:inputType="textPersonName"
                android:textColor="@android:color/white"
                android:textColorHint="@android:color/darker_gray" />

            <Button
                android:id="@+id/deleteButton"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="@string/deleteButton" />

        </TableRow>

        <TableRow
            android:layout_width="match_parent"
            android:layout_height="match_parent">
        </TableRow>

        <TableRow
            android:layout_width="match_parent"
            android:layout_height="match_parent">

        </TableRow>

        <TableRow
            android:layout_width="match_parent"
            android:layout_height="match_parent" >

            <ScrollView
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:layout_span="5">

                <TextView
                    android:id="@+id/dataDisplayTextView"
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:textColor="@android:color/white" />
            </ScrollView>

        </TableRow>

    </TableLayout>

</androidx.constraintlayout.widget.ConstraintLayout>
```