ГУАП

КАФЕДРА № 42

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ _____

ПРЕПОДАВАТЕЛЬ

старший преподаватель				С. Ю. Гуков
должность, уч. степень, звание		подпись, дата		инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №1

Введение в Android разработку

Вариант 9

по курсу: Разработка мобильных приложений. Разработка мобильных приложений на Kotlin

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. №	4128				В. А. Воробьев
			подпись, дата		инициалы, фамилия

Санкт-Петербург 2024

СОДЕРЖАНИЕ

1 Постановка задачи

Цель работы: получить практические навыки разработки простых мобильных приложений с использованием Android SDK.

1.1 Задание

Разработайте калькулятор для арифметических операций в соответствии с вариантом. Интерфейс должен быть интуитивно понятным, результат вычислений должен выводиться пользователю в Activity, все цифры и математические операции должны иметь отдельные кнопки.

Чтобы подобрать вариант необходимо найти число N = (номер группы + номер студента в журнале) mod количество_вариантов + 1. где mod – остаток от деления, N - номер варианта.

Вариант $= 1 + (4128 + 5) \bmod 15 = 9$

Преобразователь энергии (не менее 10 направлений перевода)

2 Выполнение работы

Для выполнения работы был выбран фреймворк `Jetpack Compose` в виду простоты его использования.

Итоговая структура проекта изображена рисунке 2.1

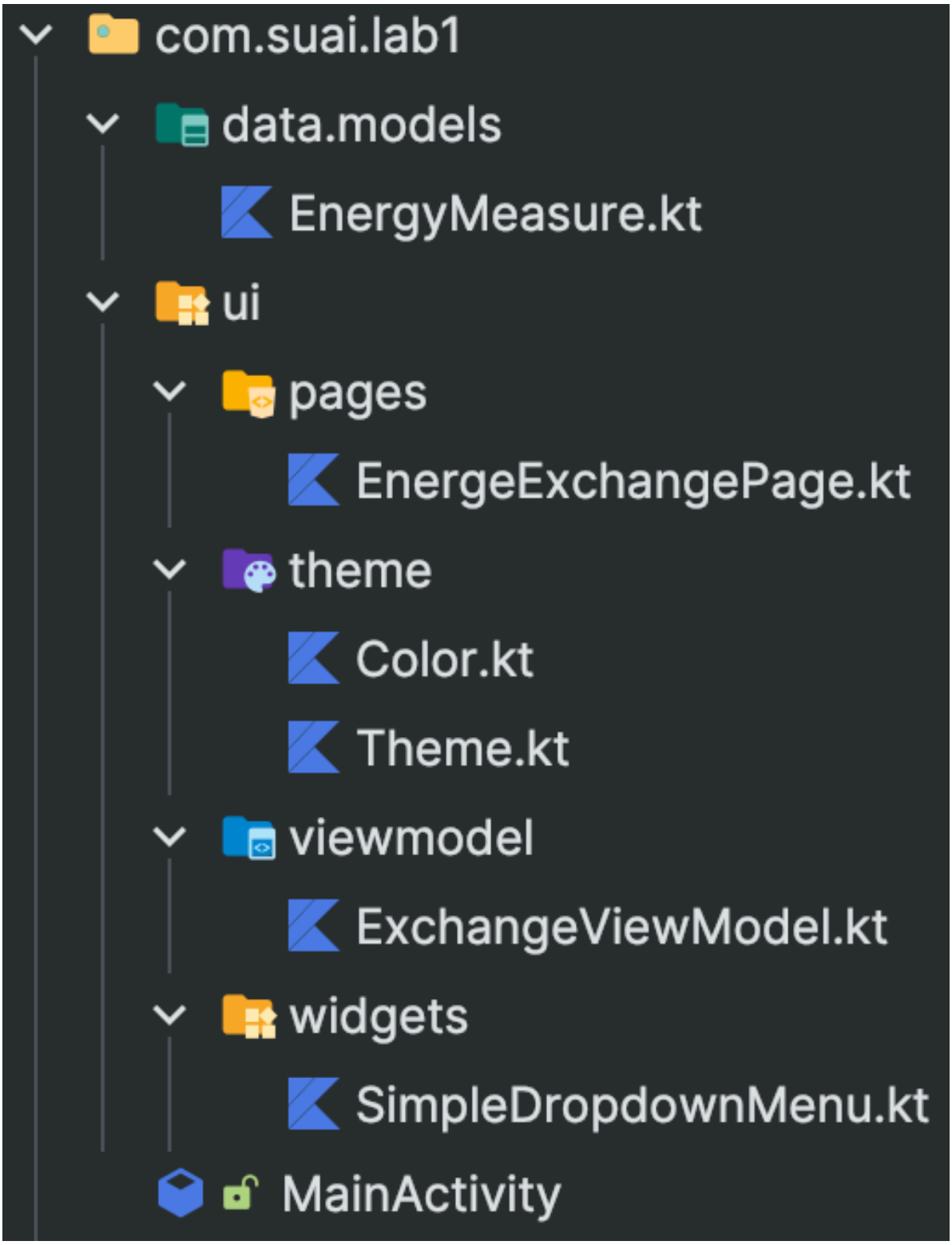

Рисунок 2.1 - Структура проекта

Код итоговый проект доступен в приложении и на GitHub.

Бизнес логика преобразования энергии находится в файле `EnergyMeasure.kt`. Главная для понимания часть представлена на листинге ниже.

```
val energyMeasurementRepository: List<EnergyMeasure> = listOf(
    EnergyMeasure("Joule", 1.0),
    EnergyMeasure("KiloJoule", 1000.0),
    EnergyMeasure("cal", 4.1868),
    EnergyMeasure("kcal", 4186.8),
    EnergyMeasure("Wh", 3600.0),
    EnergyMeasure("Ws", 1.0),
    EnergyMeasure("kWh", 3600000.0),
    EnergyMeasure("erg", .0000001),
    EnergyMeasure("kgf-m", 9.80665),
    EnergyMeasure("tm", 105506000.0),
)

data class EnergyMeasure(val name: String, val joule: Double) {
    fun convertTo(count: Double, output: EnergyMeasure): Double {
        return count * this.joule / output.joule
    }

    override fun toString() = name
}
```

Затем эта модель используется в `ExchangeViewModel`.

```
data class ExchangeState(
    val inputMeasure: EnergyMeasure,
    val inputCount: Double,
    val outputMeasure: EnergyMeasure,
) {
    val outputCount: Double = inputMeasure.convertTo(
        inputCount, outputMeasure)
}

class ExchangeViewModel : ViewModel() {
    private val _state = MutableStateFlow(
        ExchangeState(
```

```
12              inputMeasure = energyMeasurementRepository[0],
13              inputCount = 0.0,
14              outputMeasure = energyMeasurementRepository[1],
15          )
16      )
17      val state = _state.asStateFlow()
18
19      fun setInputMeasure(measure: EnergyMeasure) {
20          _state.update { state ->
21              state.copy(inputMeasure = measure)
22          }
23      }
24
25      fun setOutputMeasure(measure: EnergyMeasure) {
26          _state.update { state ->
27              state.copy(outputMeasure = measure)
28          }
29      }
30
31      fun setInputCount(count: Double) {
32          _state.update { state ->
33              state.copy(inputCount = count)
34          }
35      }
36  }
```

2.1 Демонстрация работы

На рисунках 2.2 - 2.5 представлен результат работы приложения.

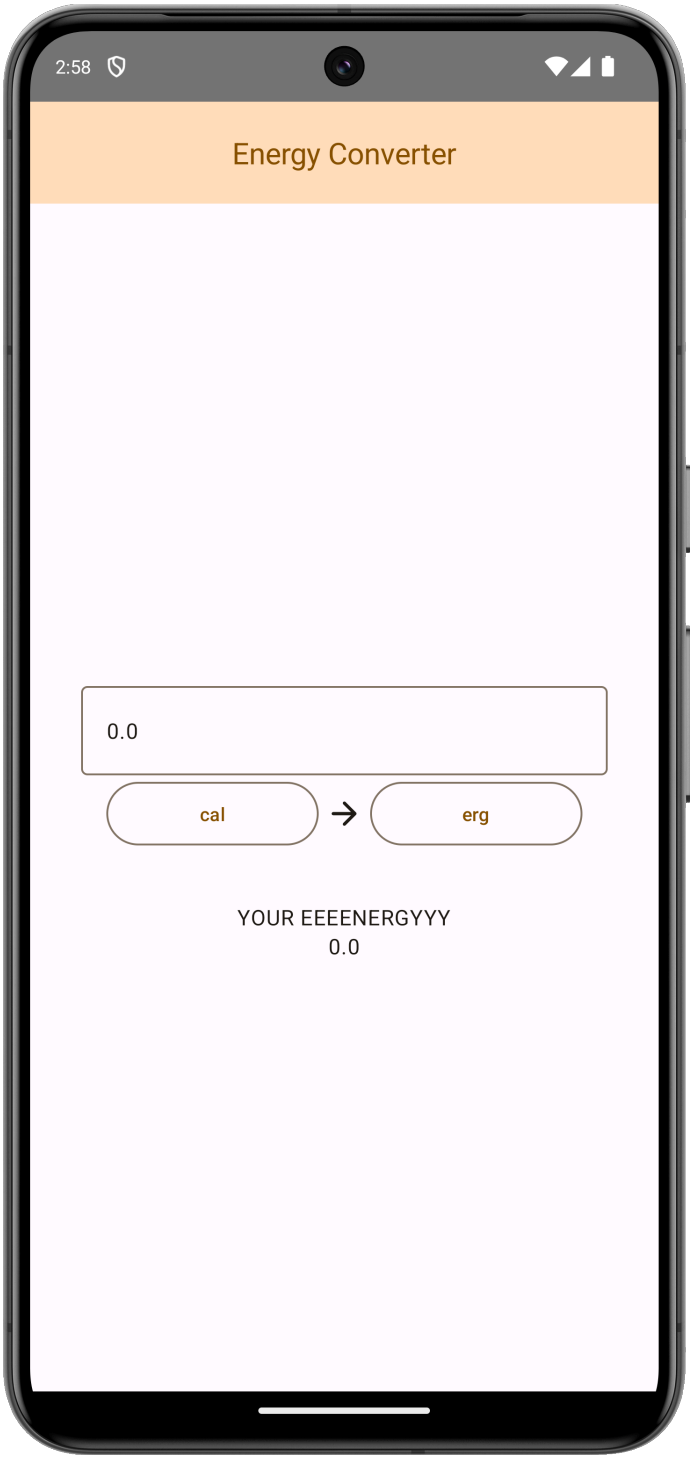

Рисунок 2.2 - Начальный экран

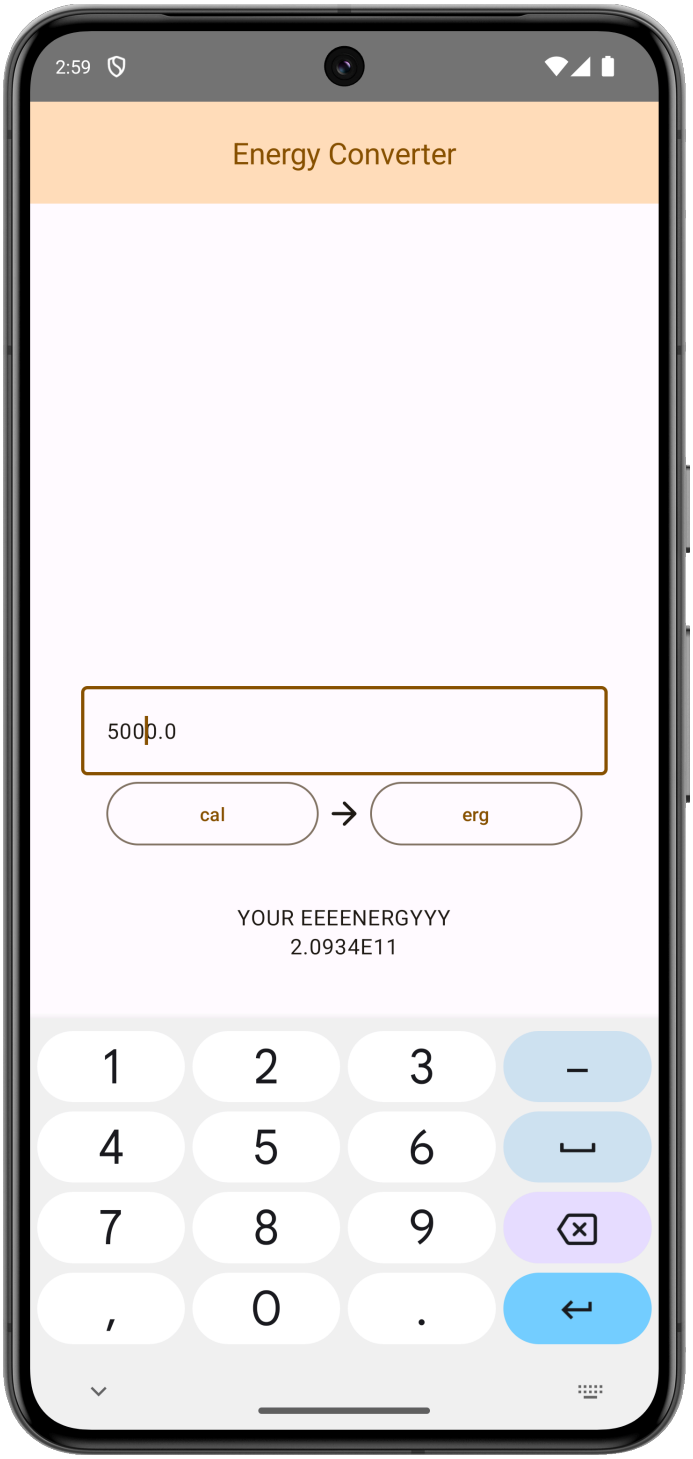

Рисунок 2.3 - Ввод энергии

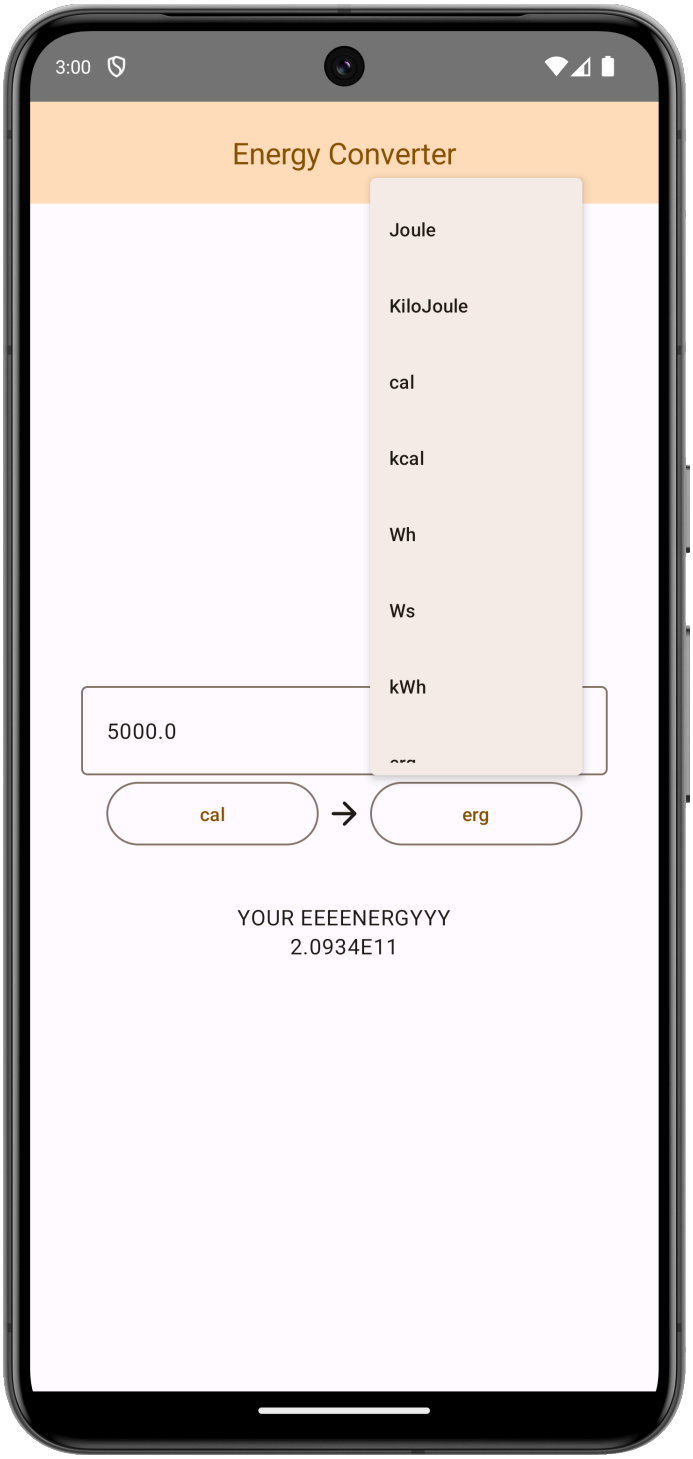

Рисунок 2.4 - Выбор единицы измерения

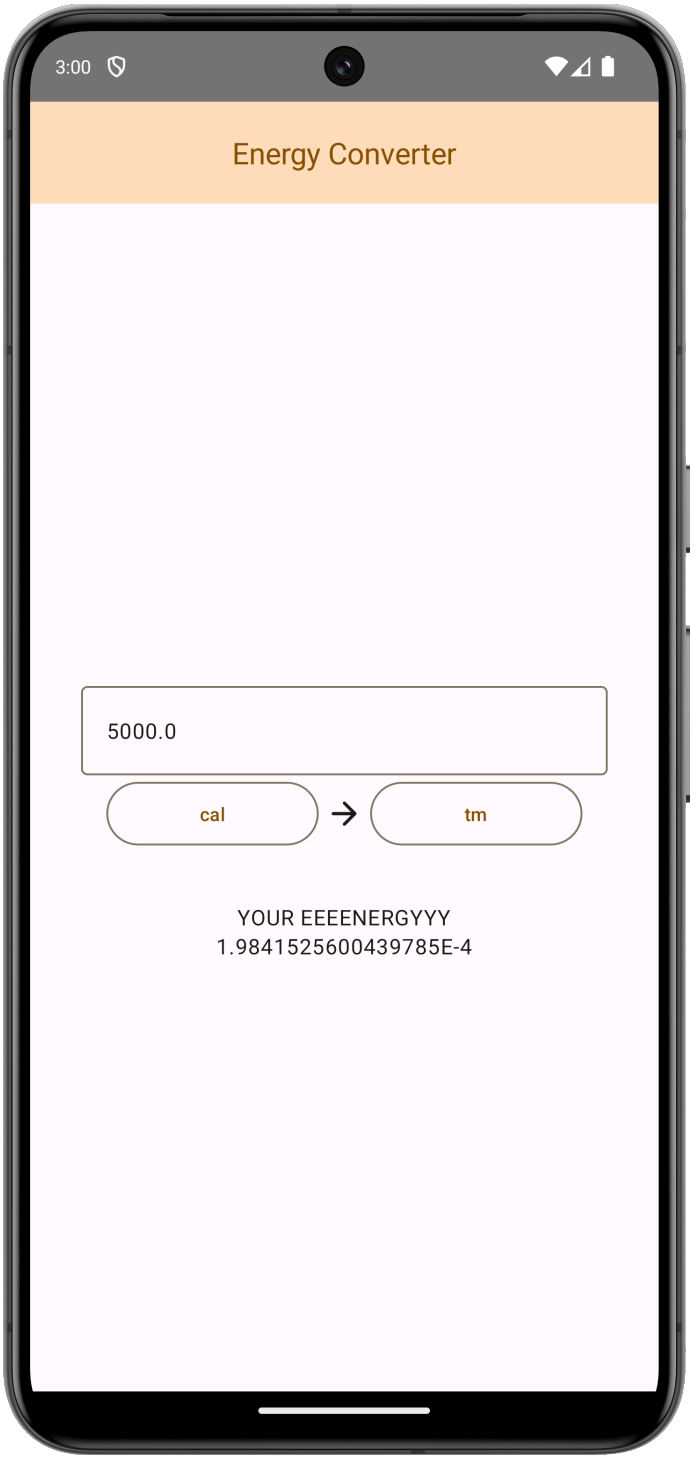

Рисунок 2.5 - Смена единицы измерения

3 Вывод

В результате выполнения работы были получены практические навыки разработки мобильных приложений с использованием Android SDK. Для реализации поставленной задачи был выбран фреймворк Jetpack Compose, обоснованный своей простотой использования.

Демонстрация работы приложения подчеркивает интуитивную понятность интерфейса. Начиная с открытия приложения, пользователь может вводить количество энергии и выбирать единицы измерения как входных, так и выходных данных.

Полученные навыки разработки мобильных приложений, особенно с использованием Jetpack Compose, представляют собой ценный опыт. Работа с библиотеками, архитектурные решения, и управление состоянием в приложении - все это важные аспекты, которые были освоены в ходе выполнения данной задачи. Эти навыки могут быть успешно применены в будущих проектах, требующих разработки мобильных приложений под Android.

Листинг

```
ExchangeViewModel.kt
package com.suai.lab1.ui.viewmodel

import androidx.lifecycle.ViewModel
import com.suai.lab1.data.models.EnergyMeasure
import com.suai.lab1.data.models.energyMeasurementRepository
import kotlinx.coroutines.flow.MutableStateFlow
import kotlinx.coroutines.flow.asStateFlow
import kotlinx.coroutines.flow.update

data class ExchangeState(
    val inputMeasure: EnergyMeasure,
    val inputCount: Double,
    val outputMeasure: EnergyMeasure,
) {
    val outputCount: Double = inputMeasure.convertTo(
       inputCount, outputMeasure)
}

class ExchangeViewModel : ViewModel() {
    private val _state = MutableStateFlow(
        ExchangeState(
            inputMeasure = energyMeasurementRepository[0],
            inputCount = 0.0,
            outputMeasure = energyMeasurementRepository[1],
        )
    )
    val state = _state.asStateFlow()

    fun setInputMeasure(measure: EnergyMeasure) {
        _state.update { state ->
            state.copy(inputMeasure = measure)
        }
    }

    fun setOutputMeasure(measure: EnergyMeasure) {
        _state.update { state ->
            state.copy(outputMeasure = measure)
        }
    }
```

```
40
41      fun setInputCount(count: Double) {
42          _state.update { state ->
43              state.copy(inputCount = count)
44          }
45      }
46  }
47  Color.kt
48  package com.example.compose
49
50  import androidx.compose.ui.graphics.Color
51
52  val md_theme_light_primary = Color(0xFF875200)
53  val md_theme_light_onPrimary = Color(0xFFFFFFFF)
54  val md_theme_light_primaryContainer = Color(0xFFFFDDBA)
55  val md_theme_light_onPrimaryContainer = Color(0xFF2B1700)
56  val md_theme_light_secondary = Color(0xFF715A41)
57  val md_theme_light_onSecondary = Color(0xFFFFFFFF)
58  val md_theme_light_secondaryContainer = Color(0xFFFDDDBD)
59  val md_theme_light_onSecondaryContainer = Color(0xFF281805)
60  val md_theme_light_tertiary = Color(0xFF8F4E00)
61  val md_theme_light_onTertiary = Color(0xFFFFFFFF)
62  val md_theme_light_tertiaryContainer = Color(0xFFFFDCC1)
63  val md_theme_light_onTertiaryContainer = Color(0xFF2E1500)
64  val md_theme_light_error = Color(0xFFBA1A1A)
65  val md_theme_light_errorContainer = Color(0xFFFFDAD6)
66  val md_theme_light_onError = Color(0xFFFFFFFF)
67  val md_theme_light_onErrorContainer = Color(0xFF410002)
68  val md_theme_light_background = Color(0xFFFFFBFF)
69  val md_theme_light_onBackground = Color(0xFF1F1B16)
70  val md_theme_light_surface = Color(0xFFFFFBFF)
71  val md_theme_light_onSurface = Color(0xFF1F1B16)
72  val md_theme_light_surfaceVariant = Color(0xFFF1DFD0)
73  val md_theme_light_onSurfaceVariant = Color(0xFF50453A)
74  val md_theme_light_outline = Color(0xFF827568)
75  val md_theme_light_inverseOnSurface = Color(0xFFFAEFE7)
76  val md_theme_light_inverseSurface = Color(0xFF352F2A)
77  val md_theme_light_inversePrimary = Color(0xFFFFB866)
78  val md_theme_light_shadow = Color(0xFF000000)
79  val md_theme_light_surfaceTint = Color(0xFF875200)
80  val md_theme_light_outlineVariant = Color(0xFFD4C4B5)
```

```
81  val md_theme_light_scrim = Color(0xFF000000)
82
83  val md_theme_dark_primary = Color(0xFFFFB866)
84  val md_theme_dark_onPrimary = Color(0xFF482900)
85  val md_theme_dark_primaryContainer = Color(0xFF673D00)
86  val md_theme_dark_onPrimaryContainer = Color(0xFFFFDDBA)
87  val md_theme_dark_secondary = Color(0xFFE0C1A3)
88  val md_theme_dark_onSecondary = Color(0xFF3F2D17)
89  val md_theme_dark_secondaryContainer = Color(0xFF58432C)
90  val md_theme_dark_onSecondaryContainer = Color(0xFFFDDDBD)
91  val md_theme_dark_tertiary = Color(0xFFFFB779)
92  val md_theme_dark_onTertiary = Color(0xFF4C2700)
93  val md_theme_dark_tertiaryContainer = Color(0xFF6C3A00)
94  val md_theme_dark_onTertiaryContainer = Color(0xFFFFDCC1)
95  val md_theme_dark_error = Color(0xFFFFB4AB)
96  val md_theme_dark_errorContainer = Color(0xFF93000A)
97  val md_theme_dark_onError = Color(0xFF690005)
98  val md_theme_dark_onErrorContainer = Color(0xFFFFDAD6)
99  val md_theme_dark_background = Color(0xFF1F1B16)
100 val md_theme_dark_onBackground = Color(0xFFEBE1D9)
101 val md_theme_dark_surface = Color(0xFF1F1B16)
102 val md_theme_dark_onSurface = Color(0xFFEBE1D9)
103 val md_theme_dark_surfaceVariant = Color(0xFF50453A)
104 val md_theme_dark_onSurfaceVariant = Color(0xFFD4C4B5)
105 val md_theme_dark_outline = Color(0xFF9D8E81)
106 val md_theme_dark_inverseOnSurface = Color(0xFF1F1B16)
107 val md_theme_dark_inverseSurface = Color(0xFFEBE1D9)
108 val md_theme_dark_inversePrimary = Color(0xFF875200)
109 val md_theme_dark_shadow = Color(0xFF000000)
110 val md_theme_dark_surfaceTint = Color(0xFFFFB866)
111 val md_theme_dark_outlineVariant = Color(0xFF50453A)
112 val md_theme_dark_scrim = Color(0xFF000000)
113
114
115 val seed = Color(0xFFFFA52A)
116
117 Theme.kt
118 package com.suai.lab1.ui.theme
119
120 import androidx.compose.foundation.isSystemInDarkTheme
121 import androidx.compose.material3.MaterialTheme
```

```
122 import androidx.compose.material3.darkColorScheme
123 import androidx.compose.material3.lightColorScheme
124 import androidx.compose.runtime.Composable
125 import com.example.compose.md_theme_dark_background
126 import com.example.compose.md_theme_dark_error
127 import com.example.compose.md_theme_dark_errorContainer
128 import com.example.compose.md_theme_dark_inverseOnSurface
129 import com.example.compose.md_theme_dark_inversePrimary
130 import com.example.compose.md_theme_dark_inverseSurface
131 import com.example.compose.md_theme_dark_onBackground
132 import com.example.compose.md_theme_dark_onError
133 import com.example.compose.md_theme_dark_onErrorContainer
134 import com.example.compose.md_theme_dark_onPrimary
135 import com.example.compose.md_theme_dark_onPrimaryContainer
136 import com.example.compose.md_theme_dark_onSecondary
137 import com.example.compose.md_theme_dark_onSecondaryContainer
138 import com.example.compose.md_theme_dark_onSurface
139 import com.example.compose.md_theme_dark_onSurfaceVariant
140 import com.example.compose.md_theme_dark_onTertiary
141 import com.example.compose.md_theme_dark_onTertiaryContainer
142 import com.example.compose.md_theme_dark_outline
143 import com.example.compose.md_theme_dark_outlineVariant
144 import com.example.compose.md_theme_dark_primary
145 import com.example.compose.md_theme_dark_primaryContainer
146 import com.example.compose.md_theme_dark_scrim
147 import com.example.compose.md_theme_dark_secondary
148 import com.example.compose.md_theme_dark_secondaryContainer
149 import com.example.compose.md_theme_dark_surface
150 import com.example.compose.md_theme_dark_surfaceTint
151 import com.example.compose.md_theme_dark_surfaceVariant
152 import com.example.compose.md_theme_dark_tertiary
153 import com.example.compose.md_theme_dark_tertiaryContainer
154 import com.example.compose.md_theme_light_background
155 import com.example.compose.md_theme_light_error
156 import com.example.compose.md_theme_light_errorContainer
157 import com.example.compose.md_theme_light_inverseOnSurface
158 import com.example.compose.md_theme_light_inversePrimary
159 import com.example.compose.md_theme_light_inverseSurface
160 import com.example.compose.md_theme_light_onBackground
161 import com.example.compose.md_theme_light_onError
162 import com.example.compose.md_theme_light_onErrorContainer
```

```
163 import com.example.compose.md_theme_light_onPrimary
164 import com.example.compose.md_theme_light_onPrimaryContainer
165 import com.example.compose.md_theme_light_onSecondary
166 import com.example.compose.
       md_theme_light_onSecondaryContainer
167 import com.example.compose.md_theme_light_onSurface
168 import com.example.compose.md_theme_light_onSurfaceVariant
169 import com.example.compose.md_theme_light_onTertiary
170 import com.example.compose.md_theme_light_onTertiaryContainer
171 import com.example.compose.md_theme_light_outline
172 import com.example.compose.md_theme_light_outlineVariant
173 import com.example.compose.md_theme_light_primary
174 import com.example.compose.md_theme_light_primaryContainer
175 import com.example.compose.md_theme_light_scrim
176 import com.example.compose.md_theme_light_secondary
177 import com.example.compose.md_theme_light_secondaryContainer
178 import com.example.compose.md_theme_light_surface
179 import com.example.compose.md_theme_light_surfaceTint
180 import com.example.compose.md_theme_light_surfaceVariant
181 import com.example.compose.md_theme_light_tertiary
182 import com.example.compose.md_theme_light_tertiaryContainer
183
184
185 private val LightColors = lightColorScheme(
186     primary = md_theme_light_primary,
187     onPrimary = md_theme_light_onPrimary,
188     primaryContainer = md_theme_light_primaryContainer,
189     onPrimaryContainer = md_theme_light_onPrimaryContainer,
190     secondary = md_theme_light_secondary,
191     onSecondary = md_theme_light_onSecondary,
192     secondaryContainer = md_theme_light_secondaryContainer,
193     onSecondaryContainer =
          md_theme_light_onSecondaryContainer,
194     tertiary = md_theme_light_tertiary,
195     onTertiary = md_theme_light_onTertiary,
196     tertiaryContainer = md_theme_light_tertiaryContainer,
197     onTertiaryContainer = md_theme_light_onTertiaryContainer,
198     error = md_theme_light_error,
199     errorContainer = md_theme_light_errorContainer,
200     onError = md_theme_light_onError,
201     onErrorContainer = md_theme_light_onErrorContainer,
```

```
202         background = md_theme_light_background,
203         onBackground = md_theme_light_onBackground,
204         surface = md_theme_light_surface,
205         onSurface = md_theme_light_onSurface,
206         surfaceVariant = md_theme_light_surfaceVariant,
207         onSurfaceVariant = md_theme_light_onSurfaceVariant,
208         outline = md_theme_light_outline,
209         inverseOnSurface = md_theme_light_inverseOnSurface,
210         inverseSurface = md_theme_light_inverseSurface,
211         inversePrimary = md_theme_light_inversePrimary,
212         surfaceTint = md_theme_light_surfaceTint,
213         outlineVariant = md_theme_light_outlineVariant,
214         scrim = md_theme_light_scrim,
215     )
216
217
218     private val DarkColors = darkColorScheme(
219         primary = md_theme_dark_primary,
220         onPrimary = md_theme_dark_onPrimary,
221         primaryContainer = md_theme_dark_primaryContainer,
222         onPrimaryContainer = md_theme_dark_onPrimaryContainer,
223         secondary = md_theme_dark_secondary,
224         onSecondary = md_theme_dark_onSecondary,
225         secondaryContainer = md_theme_dark_secondaryContainer,
226         onSecondaryContainer = md_theme_dark_onSecondaryContainer
              ,
227         tertiary = md_theme_dark_tertiary,
228         onTertiary = md_theme_dark_onTertiary,
229         tertiaryContainer = md_theme_dark_tertiaryContainer,
230         onTertiaryContainer = md_theme_dark_onTertiaryContainer,
231         error = md_theme_dark_error,
232         errorContainer = md_theme_dark_errorContainer,
233         onError = md_theme_dark_onError,
234         onErrorContainer = md_theme_dark_onErrorContainer,
235         background = md_theme_dark_background,
236         onBackground = md_theme_dark_onBackground,
237         surface = md_theme_dark_surface,
238         onSurface = md_theme_dark_onSurface,
239         surfaceVariant = md_theme_dark_surfaceVariant,
240         onSurfaceVariant = md_theme_dark_onSurfaceVariant,
241         outline = md_theme_dark_outline,
```

```
242         inverseOnSurface = md_theme_dark_inverseOnSurface,
243         inverseSurface = md_theme_dark_inverseSurface,
244         inversePrimary = md_theme_dark_inversePrimary,
245         surfaceTint = md_theme_dark_surfaceTint,
246         outlineVariant = md_theme_dark_outlineVariant,
247         scrim = md_theme_dark_scrim,
248     )
249
250     @Composable
251     fun AppTheme(
252         useDarkTheme: Boolean = isSystemInDarkTheme(),
253         content: @Composable() () -> Unit
254     ) {
255         val colors = if (!useDarkTheme) {
256             LightColors
257         } else {
258             DarkColors
259         }
260
261         MaterialTheme(
262             colorScheme = colors,
263             content = content
264         )
265     }
266     EnergeExchangePage.kt
267     package com.suai.lab1.ui.pages
268
269     import androidx.compose.foundation.layout.Arrangement
270     import androidx.compose.foundation.layout.Column
271     import androidx.compose.foundation.layout.
            ExperimentalLayoutApi
272     import androidx.compose.foundation.layout.FlowRow
273     import androidx.compose.foundation.layout.Spacer
274     import androidx.compose.foundation.layout.fillMaxSize
275     import androidx.compose.foundation.layout.fillMaxWidth
276     import androidx.compose.foundation.layout.height
277     import androidx.compose.foundation.layout.padding
278     import androidx.compose.foundation.text.KeyboardOptions
279     import androidx.compose.material.icons.Icons
280     import androidx.compose.material.icons.rounded.ArrowForward
281     import androidx.compose.material3.CenterAlignedTopAppBar
```

```
282 import androidx.compose.material3.ExperimentalMaterial3Api
283 import androidx.compose.material3.Icon
284 import androidx.compose.material3.MaterialTheme
285 import androidx.compose.material3.OutlinedTextField
286 import androidx.compose.material3.Scaffold
287 import androidx.compose.material3.Text
288 import androidx.compose.material3.TopAppBarDefaults
289 import androidx.compose.runtime.Composable
290 import androidx.compose.runtime.collectAsState
291 import androidx.compose.runtime.getValue
292 import androidx.compose.ui.Alignment
293 import androidx.compose.ui.Modifier
294 import androidx.compose.ui.platform.LocalFocusManager
295 import androidx.compose.ui.text.input.KeyboardType
296 import androidx.compose.ui.tooling.preview.Devices
297 import androidx.compose.ui.tooling.preview.Preview
298 import androidx.compose.ui.unit.dp
299 import com.suai.lab1.data.models.energyMeasurementRepository
300 import com.suai.lab1.ui.viewmodel.ExchangeViewModel
301 import com.suai.lab1.ui.theme.AppTheme
302 import com.suai.lab1.ui.widgets.SimpleDropdownMenu
303
304 @OptIn(ExperimentalLayoutApi::class, ExperimentalMaterial3Api
       ::class)
305 @Composable
306 fun EnergyExchangePage(
307     exchangeViewModel: ExchangeViewModel = ExchangeViewModel
        ()
308 ) {
309     val space = @Composable {
310         Spacer(
311             modifier = Modifier.height(32.dp)
312         )
313     }
314     val measurements = energyMeasurementRepository
315     val exchangeState by exchangeViewModel.state.
        collectAsState()
316     val focusManager = LocalFocusManager.current
317
318     Scaffold(
319         topBar = {
```

```
320             CenterAlignedTopAppBar(
321                 title = { Text("Energy Converter") },
322                 colors = TopAppBarDefaults.
                        centerAlignedTopAppBarColors(
323                     containerColor = MaterialTheme.
                            colorScheme.primaryContainer,
324                     titleContentColor = MaterialTheme.
                            colorScheme.primary,
325                 ),
326             )
327         }
328     ) { innerPadding ->
329         Column(
330             modifier = Modifier
331                 .padding(innerPadding)
332                 .padding(32.dp)
333                 .fillMaxSize(),
334             horizontalAlignment = Alignment.
                    CenterHorizontally,
335             verticalArrangement = Arrangement.Center,
336         ) {
337             space()
338             OutlinedTextField(
339                 value = exchangeState.inputCount.toBigDecimal
                        ().toPlainString(),
340                 onValueChange = {
341                     if (it.replace("\n", "q").toDoubleOrNull
                            () == null) {
342                         focusManager.clearFocus()
343                         return@OutlinedTextField
344                     }
345                     exchangeViewModel.setInputCount(it.
                            toDouble())
346                     println(exchangeState.outputCount)
347                 },
348                 keyboardOptions = KeyboardOptions(
349                     keyboardType = KeyboardType.Number,
350                 ),
351                 modifier = Modifier.fillMaxWidth()
352             )
353             FlowRow(
```

```
354                     modifier = Modifier.padding(horizontal = 16.
                            dp),
355                     horizontalArrangement = Arrangement.spacedBy
                            (4.dp)
356                 ) {
357                     SimpleDropdownMenu(
358                         item = exchangeState.inputMeasure,
359                         options = measurements,
360                         onOptionSelected = exchangeViewModel::
                                setInputMeasure,
361                         modifier = Modifier.weight(1f)
362                     )
363                     Icon(
364                         Icons.Rounded.ArrowForward,
365                         contentDescription = "",
366                         modifier = Modifier.align(Alignment.
                                CenterVertically)
367                     )
368                     SimpleDropdownMenu(
369                         item = exchangeState.outputMeasure,
370                         options = measurements,
371                         onOptionSelected = exchangeViewModel::
                                setOutputMeasure,
372                         modifier = Modifier.weight(1f)
373                     )
374
375                 }
376                 space()
377                 Text(text = "YOUR EEEENERGYYY")
378                 Text(exchangeState.outputCount.toString())
379             }
380         }
381 }
382
383 @Preview(showBackground = true, device = Devices.PIXEL_4)
384 @Composable
385 fun EnergyExchangePagePreview() {
386     AppTheme {
387         EnergyExchangePage()
388     }
389 }
```

```
SimpleDropdownMenu.kt
@file:OptIn(ExperimentalMaterial3Api::class)

package com.suai.lab1.ui.widgets

import androidx.compose.foundation.layout.fillMaxWidth
import androidx.compose.material3.DropdownMenuItem
import androidx.compose.material3.ExperimentalMaterial3Api
import androidx.compose.material3.ExposedDropdownMenuBox
import androidx.compose.material3.OutlinedButton
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.runtime.getValue
import androidx.compose.runtime.mutableStateOf
import androidx.compose.runtime.remember
import androidx.compose.runtime.setValue
import androidx.compose.ui.Modifier
import androidx.compose.ui.tooling.preview.Preview

@Composable
fun <T> SimpleDropdownMenu(
    item: T,
    options: List<T>,
    onOptionSelected: (T) -> Unit,
    modifier: Modifier = Modifier
) {
    var expanded by remember { mutableStateOf(false) }

    ExposedDropdownMenuBox(
        expanded = expanded,
        onExpandedChange = {
            expanded = !expanded
        },
        modifier = modifier.fillMaxWidth(),
    ) {
        OutlinedButton(
            onClick = {},
            modifier = Modifier
                .menuAnchor()
                .fillMaxWidth(),
        ) {
```

```
431             Text(text = item.toString())
432         }
433         ExposedDropdownMenu(expanded = expanded,
                onDismissRequest = {
434             expanded = false
435         }) {
436             options.forEach { selectionOption ->
437                 DropdownMenuItem(
438                     text = { Text(text = selectionOption.
                        toString()) },
439                     onClick = {
440                         onOptionSelected(selectionOption)
441                         expanded = false
442                     },
443                 )
444             }
445         }
446     }
447 }
448
449 @Preview(widthDp = 200, heightDp = 400)
450 @Composable
451 fun SimplePreview() {
452     val options = listOf("Option 1", "Option 2", "Option 3",
        "Option 4")
453     var selected by remember {
454         mutableStateOf(options[0])
455     }
456     SimpleDropdownMenu(item = selected, options = options,
        onOptionSelected = { selected = it })
457 }
458
459 MainActivity.kt
460 package com.suai.lab1
461
462 import android.os.Bundle
463 import androidx.activity.ComponentActivity
464 import androidx.activity.compose.setContent
465 import androidx.activity.enableEdgeToEdge
466 import com.suai.lab1.ui.pages.EnergyExchangePage
467 import com.suai.lab1.ui.theme.AppTheme
```

```
468
469 class MainActivity : ComponentActivity() {
470     override fun onCreate(savedInstanceState: Bundle?) {
471         super.onCreate(savedInstanceState)
472         enableEdgeToEdge()
473         setContent {
474             AppTheme {
475                 EnergyExchangePage()
476             }
477         }
478     }
479 }
480 EnergyMeasure.kt
481 package com.suai.lab1.data.models
482
483 val energyMeasurementRepository: List<EnergyMeasure> = listOf
    (
484     EnergyMeasure("Joule", 1.0),
485     EnergyMeasure("KiloJoule", 1000.0),
486     EnergyMeasure("cal", 4.1868),
487     EnergyMeasure("kcal", 4186.8),
488     EnergyMeasure("Wh", 3600.0),
489     EnergyMeasure("Ws", 1.0),
490     EnergyMeasure("kWh", 3600000.0),
491     EnergyMeasure("erg", .0000001),
492     EnergyMeasure("kgf-m", 9.80665),
493     EnergyMeasure("tm", 105506000.0),
494 )
495
496 data class EnergyMeasure(val name: String, val joule: Double)
    {
497     fun convertTo(count: Double, output: EnergyMeasure):
        Double {
498         return count * this.joule / output.joule
499     }
500
501     override fun toString() = name
502 }
```