Міністерство освіти і науки України  
Національний технічний університет України

«Київський політехнічний інститут ім. І. Сікорського»

Кафедра інженерії програмного забезпечення в енергетиці

Лабораторна робота № 4

з курсу: “Розробка програмного забезпечення мобільних пристроїв”

**Виконав:**  
студент 4-го курсу,  
групи ТВ-11

Гудзовський Марк Юрійович

Посилання на GitHub репозиторій:

<https://github.com/Chivas1717/kpi-kotlin-lab-4>

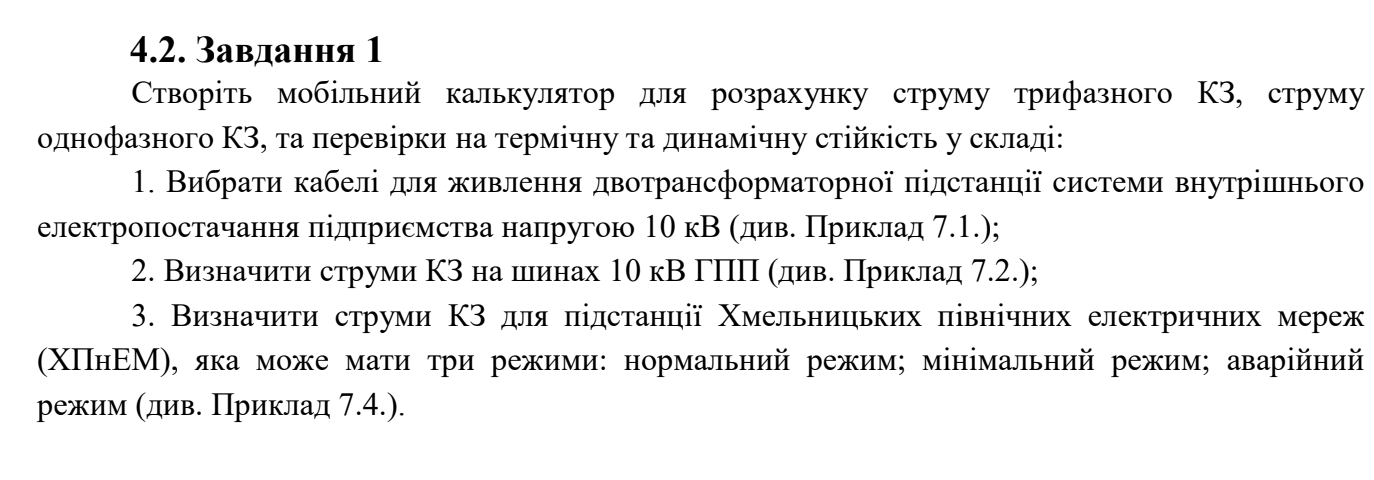
**Перевірив:**

Недашківський О.Л.

Київ 2024/2025

Лабораторна робота № 4

**Завдання:**



**Хід виконання:**

Перед тим, як обчислювати значення функії/шукати розв'язки рівняння, потрібно ініцілізувати змінні та зчитати їх з інпутів в додатку:

MainActivity.kt

// MainActivity.kt

package com.example.kpikotlinlab4

import android.os.Bundle

import androidx.activity.ComponentActivity

import androidx.activity.compose.setContent

import androidx.compose.foundation.layout.\*

import androidx.compose.material3.\*

import androidx.compose.runtime.\*

import androidx.compose.ui.Modifier

import androidx.compose.ui.tooling.preview.Preview

import androidx.compose.ui.unit.dp

import kotlin.math.exp

import kotlin.math.pow

import kotlin.math.sqrt

import kotlin.math.*PI*

import androidx.compose.foundation.verticalScroll

import androidx.compose.foundation.rememberScrollState

import androidx.compose.ui.platform.*LocalFocusManager*

import androidx.compose.foundation.clickable

import androidx.compose.runtime.Composable

class MainActivity : ComponentActivity() {

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

*setContent* **{**

MainScreen()

**}**

}

}

@Composable

fun MainScreen() {

// стейт для індексу обраного табу

var selectedTabIndex by remember **{** *mutableStateOf*(0) **}**

// визначаємо таби

Scaffold(

topBar = **{**

TabRow(

selectedTabIndex = selectedTabIndex,

modifier = Modifier.*fillMaxWidth*()

) **{**

// Define tabs

Tab(

selected = selectedTabIndex == 0,

onClick = **{** selectedTabIndex = 0 **}**,

text = **{** Text("Калькулятор 1") **}**

)

Tab(

selected = selectedTabIndex == 1,

onClick = **{** selectedTabIndex = 1 **}**,

text = **{** Text("Калькулятор 2") **}**

)

Tab(

selected = selectedTabIndex == 2,

onClick = **{** selectedTabIndex = 2 **}**,

text = **{** Text("Калькулятор 3") **}**

)

**}**

**}**

) **{** innerPadding **->**

// показуємо composable в залежності від обраного табу

when (selectedTabIndex) {

0 -> CableCalculatorApp(modifier = Modifier.*padding*(innerPadding))

1 -> CurrentCalculatorApp(modifier = Modifier.*padding*(innerPadding))

2 -> Calculator3App(modifier = Modifier.*padding*(innerPadding))

}

**}**

}

@Preview(showBackground = true)

@Composable

fun MainScreenAppPreview() {

MainScreen()

}

В цьому файлі виконуються навігація та ініціалізація усіх калькуляторів.

**Завдання 1**

CalculatorOne.kt

package com.example.kpikotlinlab4

import androidx.compose.foundation.clickable

import androidx.compose.foundation.layout.Column

import androidx.compose.foundation.layout.Spacer

import androidx.compose.foundation.layout.fillMaxSize

import androidx.compose.foundation.layout.fillMaxWidth

import androidx.compose.foundation.layout.height

import androidx.compose.foundation.layout.padding

import androidx.compose.foundation.rememberScrollState

import androidx.compose.foundation.verticalScroll

import androidx.compose.material3.Button

import androidx.compose.material3.DropdownMenu

import androidx.compose.material3.DropdownMenuItem

import androidx.compose.material3.MaterialTheme

import androidx.compose.material3.OutlinedTextField

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.platform.*LocalFocusManager*

import androidx.compose.ui.unit.dp

import kotlin.math.sqrt

import androidx.compose.ui.Modifier

@Composable

fun CableCalculatorApp(modifier: Modifier = Modifier) {

val focusManager = *LocalFocusManager*.current

// визначаю змінні

var Unom by remember **{** *mutableStateOf*("10") **}**

var Ik by remember **{** *mutableStateOf*("2.5") **}**

var Tf by remember **{** *mutableStateOf*("2.5") **}**

var Potuzhnist\_TP by remember **{** *mutableStateOf*("2000") **}**

var Sm by remember **{** *mutableStateOf*("1300") **}**

var Tm by remember **{** *mutableStateOf*("4000") **}**

var Ct by remember **{** *mutableStateOf*("92") **}**

var Im by remember **{** *mutableStateOf*("") **}**

var Impa by remember **{** *mutableStateOf*("") **}**

var Sek by remember **{** *mutableStateOf*("") **}**

var s\_smin by remember **{** *mutableStateOf*("") **}**

var conductorTypeExpanded by remember **{** *mutableStateOf*(false) **}**

var conductorType by remember **{** *mutableStateOf*("Тип провідника:") **}**

val conductorTypeOptions = *listOf*(

"Неізольовані проводи та шини",

"Кабелі з паперовою і проводи з гумовою та полівінілхлоридною ізоляцією з жилами",

"Кабелі з гумовою та пластмасовою ізоляцією з жилами"

)

var conductorMaterialExpanded by remember **{** *mutableStateOf*(false) **}**

var conductorMaterial by remember **{** *mutableStateOf*("Матеріал провідника") **}**

val conductorMaterialOptions = *listOf*(

"мідні",

"алюмінієві"

)

// розмітка застосунку

Column(

modifier = Modifier

.*fillMaxSize*()

.*padding*(16.*dp*)

.*verticalScroll*(rememberScrollState())

) **{**

Spacer(modifier = Modifier.*height*(32.*dp*))

OutlinedTextField(

value = Unom,

onValueChange = **{** Unom = **it }**,

label = **{** Text("Unom") **}**,

modifier = Modifier

.*fillMaxWidth*()

.*clickable* **{** focusManager.clearFocus() **}**

)

Spacer(modifier = Modifier.*height*(8.*dp*))

OutlinedTextField(

value = Ik,

onValueChange = **{** Ik = **it }**,

label = **{** Text("Ik") **}**,

modifier = Modifier

.*fillMaxWidth*()

.*clickable* **{** focusManager.clearFocus() **}**

)

Spacer(modifier = Modifier.*height*(8.*dp*))

OutlinedTextField(

value = Tf,

onValueChange = **{** Tf = **it }**,

label = **{** Text("Tf") **}**,

modifier = Modifier

.*fillMaxWidth*()

.*clickable* **{** focusManager.clearFocus() **}**

)

Spacer(modifier = Modifier.*height*(8.*dp*))

OutlinedTextField(

value = Potuzhnist\_TP,

onValueChange = **{** Potuzhnist\_TP = **it }**,

label = **{** Text("Potuzhnist\_TP") **}**,

modifier = Modifier

.*fillMaxWidth*()

.*clickable* **{** focusManager.clearFocus() **}**

)

Spacer(modifier = Modifier.*height*(8.*dp*))

OutlinedTextField(

value = Sm,

onValueChange = **{** Sm = **it }**,

label = **{** Text("Sm") **}**,

modifier = Modifier

.*fillMaxWidth*()

.*clickable* **{** focusManager.clearFocus() **}**

)

Spacer(modifier = Modifier.*height*(8.*dp*))

OutlinedTextField(

value = Tm,

onValueChange = **{** Tm = **it }**,

label = **{** Text("Tm") **}**,

modifier = Modifier

.*fillMaxWidth*()

.*clickable* **{** focusManager.clearFocus() **}**

)

Spacer(modifier = Modifier.*height*(8.*dp*))

// Dropdown menu

Button(onClick = **{** conductorTypeExpanded = true **}**) **{**

Text(conductorType)

**}**

DropdownMenu(

expanded = conductorTypeExpanded,

onDismissRequest = **{** conductorTypeExpanded = false **}**

) **{**

conductorTypeOptions.*forEach* **{** option **->**

DropdownMenuItem(

onClick = **{**

conductorType = option

conductorTypeExpanded = false

**}**,

text = **{** Text(option) **}**

)

**}**

**}**

Spacer(modifier = Modifier.*height*(8.*dp*))

Button(onClick = **{** conductorMaterialExpanded = true **}**) **{**

Text(conductorMaterial)

**}**

DropdownMenu(

expanded = conductorMaterialExpanded,

onDismissRequest = **{** conductorMaterialExpanded = false **}**

) **{**

conductorMaterialOptions.*forEach* **{** option **->**

DropdownMenuItem(

onClick = **{**

conductorMaterial = option

conductorMaterialExpanded = false

**}**,

text = **{** Text(option) **}**

)

**}**

**}**

Spacer(modifier = Modifier.*height*(16.*dp*))

// кнопка для виконання обчислень

Button(

onClick = **{**

val Sm = Sm.*toDoubleOrNull*() ?: 0.0

val Unom = Unom.*toDoubleOrNull*() ?: 0.0

val Tm = Tm.*toDoubleOrNull*() ?: 0.0

val Ik = Ik.*toDoubleOrNull*() ?: 0.0

var \_Im = (Sm / 2.0) / (*sqrt*(3.0) \* Unom)

var \_Impa = 2 \* \_Im

val Jek = *determineJek*(conductorType, conductorMaterial, Tm)

var \_Sek = \_Im / Jek

\_Im = Math.round(\_Im \* 100) / 100.0

\_Impa = Math.round(\_Impa \* 100) / 100.0

\_Sek = Math.round(\_Sek \* 100) / 100.0

Im = \_Im.toString()

Impa = \_Impa.toString()

Sek = \_Sek.toString()

**}**,

modifier = Modifier.*fillMaxWidth*()

) **{**

Text("Обчислити")

**}**

// вивід результатів

Text(text = "Розрахунковий струм для нормального режиму: " + Im,

style = MaterialTheme.typography.bodyLarge)

Spacer(modifier = Modifier.*height*(8.*dp*))

Text(text = "Розрахунковий струм для післяаварійного режиму: " + Impa,

style = MaterialTheme.typography.bodyLarge)

Spacer(modifier = Modifier.*height*(8.*dp*))

Text(text = "Економічний переріз (S ек): " + Sek,

style = MaterialTheme.typography.bodyLarge)

Spacer(modifier = Modifier.*height*(8.*dp*))

Spacer(modifier = Modifier.*height*(8.*dp*))

OutlinedTextField(

value = Ct,

onValueChange = **{** Ct = **it }**,

label = **{** Text("Ct") **}**,

modifier = Modifier

.*fillMaxWidth*()

.*clickable* **{** focusManager.clearFocus() **}**

)

Spacer(modifier = Modifier.*height*(8.*dp*))

Button(

onClick = **{**

val Tf = Tf.*toDoubleOrNull*() ?: 0.0

val Ik = Ik.*toDoubleOrNull*() ?: 0.0

val Ct = Ct.*toDoubleOrNull*() ?: 0.0

var \_s\_smin = (Ik \* 1000 \* *sqrt*(Tf)) / Ct

\_s\_smin = Math.round(\_s\_smin \* 100) / 100.0

s\_smin = \_s\_smin.toString()

**}**,

modifier = Modifier.*fillMaxWidth*()

) **{**

Text("Обчислити переріз за термічною стійкістю до дії струмів КЗ")

**}**

Spacer(modifier = Modifier.*height*(8.*dp*))

Text(text = "Переріз за термічною стійкістю до дії струмів КЗ: " + s\_smin,

style = MaterialTheme.typography.bodyLarge)

Spacer(modifier = Modifier.*height*(48.*dp*))

// profit.toDoubleOrNull()?.let {

// Text(text = "Висновок: " + conclusion, style = MaterialTheme.typography.bodyLarge)

// }

**}**

}

fun determineJek(conductorType: String, conductorMaterial: String, Tm: Double): Double {

// Define ranges and corresponding Jek values

val jekValues = *mapOf*(

"Неізольовані проводи та шини" *to mapOf*(

"мідні" *to listOf*(1000.0 *to* 3000.0 *to* 2.5, 3000.0 *to* 5000.0 *to* 2.1, 5000.0 *to* Double.MAX\_VALUE *to* 1.8),

"алюмінієві" *to listOf*(1000.0 *to* 3000.0 *to* 1.3, 3000.0 *to* 5000.0 *to* 1.1, 5000.0 *to* Double.MAX\_VALUE *to* 1.0)

),

"Кабелі з паперовою і проводи з гумовою та полівінілхлоридною ізоляцією з жилами" *to mapOf*(

"мідні" *to listOf*(1000.0 *to* 3000.0 *to* 3.0, 3000.0 *to* 5000.0 *to* 2.5, 5000.0 *to* Double.MAX\_VALUE *to* 2.0),

"алюмінієві" *to listOf*(1000.0 *to* 3000.0 *to* 1.6, 3000.0 *to* 5000.0 *to* 1.4, 5000.0 *to* Double.MAX\_VALUE *to* 1.2)

),

"Кабелі з гумовою та пластмасовою ізоляцією з жилами" *to mapOf*(

"мідні" *to listOf*(1000.0 *to* 3000.0 *to* 3.5, 3000.0 *to* 5000.0 *to* 3.1, 5000.0 *to* Double.MAX\_VALUE *to* 2.7),

"алюмінієві" *to listOf*(1000.0 *to* 3000.0 *to* 1.9, 3000.0 *to* 5000.0 *to* 1.7, 5000.0 *to* Double.MAX\_VALUE *to* 1.6)

)

)

// Look up the Jek value based on input parameters

return jekValues[conductorType]?.get(conductorMaterial)?.*firstOrNull* **{** (range, jek) **->**

Tm >= range.first && Tm < range.second

**}**?.second ?: 0.0

}

**Завдання 2**

CalculatorTwo.kt

package com.example.kpikotlinlab4

import androidx.compose.foundation.clickable

import androidx.compose.foundation.layout.Column

import androidx.compose.foundation.layout.Spacer

import androidx.compose.foundation.layout.fillMaxSize

import androidx.compose.foundation.layout.fillMaxWidth

import androidx.compose.foundation.layout.height

import androidx.compose.foundation.layout.padding

import androidx.compose.foundation.rememberScrollState

import androidx.compose.foundation.verticalScroll

import androidx.compose.material3.Button

import androidx.compose.material3.DropdownMenu

import androidx.compose.material3.DropdownMenuItem

import androidx.compose.material3.MaterialTheme

import androidx.compose.material3.OutlinedTextField

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.platform.*LocalFocusManager*

import androidx.compose.ui.unit.dp

import kotlin.math.sqrt

import androidx.compose.ui.Modifier

@Composable

fun CurrentCalculatorApp(modifier: Modifier = Modifier) {

val focusManager = *LocalFocusManager*.current

// визначаю змінні

var Ucn by remember **{** *mutableStateOf*("10.5") **}**

var Sk by remember **{** *mutableStateOf*("200") **}**

var Uk\_perc by remember **{** *mutableStateOf*("10.5") **}**

var S\_nom\_t by remember **{** *mutableStateOf*("6.3") **}**

var Xt by remember **{** *mutableStateOf*("") **}**

var Xc by remember **{** *mutableStateOf*("") **}**

var Xe by remember **{** *mutableStateOf*("") **}**

var Ip0 by remember **{** *mutableStateOf*("") **}**

// розмітка застосунку

Column(

modifier = Modifier

.*fillMaxSize*()

.*padding*(16.*dp*)

.*verticalScroll*(rememberScrollState())

) **{**

Spacer(modifier = Modifier.*height*(32.*dp*))

OutlinedTextField(

value = Ucn,

onValueChange = **{** Ucn = **it }**,

label = **{** Text("Ucn") **}**,

modifier = Modifier

.*fillMaxWidth*()

.*clickable* **{** focusManager.clearFocus() **}**

)

Spacer(modifier = Modifier.*height*(8.*dp*))

OutlinedTextField(

value = Sk,

onValueChange = **{** Sk = **it }**,

label = **{** Text("Sk") **}**,

modifier = Modifier

.*fillMaxWidth*()

.*clickable* **{** focusManager.clearFocus() **}**

)

Spacer(modifier = Modifier.*height*(8.*dp*))

OutlinedTextField(

value = Uk\_perc,

onValueChange = **{** Uk\_perc = **it }**,

label = **{** Text("Uk\_perc") **}**,

modifier = Modifier

.*fillMaxWidth*()

.*clickable* **{** focusManager.clearFocus() **}**

)

Spacer(modifier = Modifier.*height*(8.*dp*))

OutlinedTextField(

value = S\_nom\_t,

onValueChange = **{** S\_nom\_t = **it }**,

label = **{** Text("S\_nom\_t") **}**,

modifier = Modifier

.*fillMaxWidth*()

.*clickable* **{** focusManager.clearFocus() **}**

)

Spacer(modifier = Modifier.*height*(8.*dp*))

// кнопка для виконання обчислень

Button(

onClick = **{**

val \_Ucn = Ucn.*toDoubleOrNull*() ?: 0.0

val \_Sk = Sk.*toDoubleOrNull*() ?: 0.0

val \_Uk\_perc = Uk\_perc.*toDoubleOrNull*() ?: 0.0

val \_S\_nom\_t = S\_nom\_t.*toDoubleOrNull*() ?: 0.0

var \_Xc = Math.round((\_Ucn \* \_Ucn / \_Sk) \* 100) / 100.0

var \_Xt = Math.round(((\_Uk\_perc / 100) \* (\_Ucn \* \_Ucn / \_S\_nom\_t)) \* 100) / 100.0

var \_Xe = \_Xc + \_Xt

var \_Ip0 = Math.round((\_Ucn / (*sqrt*(3.0) \* \_Xe)) \* 100) / 100.0

Xt = \_Xc.toString()

Xc = \_Xt.toString()

Xe = \_Xe.toString()

Ip0 = \_Ip0.toString()

**}**,

modifier = Modifier.*fillMaxWidth*()

) **{**

Text("Обчислити")

**}**

// вивід результатів

Text(text = "Xc: " + Xc,

style = MaterialTheme.typography.bodyLarge)

Spacer(modifier = Modifier.*height*(8.*dp*))

Text(text = "Xt: " + Xt,

style = MaterialTheme.typography.bodyLarge)

Spacer(modifier = Modifier.*height*(8.*dp*))

Text(text = "Сумарний опір точки К1: " + Xe,

style = MaterialTheme.typography.bodyLarge)

Text(text = "Початкове діюче значення струму трифазного КЗ: " + Ip0,

style = MaterialTheme.typography.bodyLarge)

Spacer(modifier = Modifier.*height*(8.*dp*))

**}**

}

**Завдання 3**

CalculatorThree.kt

package com.example.kpikotlinlab4

import android.util.Log

import androidx.compose.foundation.clickable

import androidx.compose.foundation.layout.Column

import androidx.compose.foundation.layout.Spacer

import androidx.compose.foundation.layout.fillMaxSize

import androidx.compose.foundation.layout.fillMaxWidth

import androidx.compose.foundation.layout.height

import androidx.compose.foundation.layout.padding

import androidx.compose.foundation.rememberScrollState

import androidx.compose.foundation.verticalScroll

import androidx.compose.material3.Button

import androidx.compose.material3.MaterialTheme

import androidx.compose.material3.OutlinedTextField

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.platform.*LocalFocusManager*

import androidx.compose.ui.unit.dp

import kotlin.math.sqrt

import kotlin.math.exp

import kotlin.math.ln

import androidx.compose.ui.Modifier

@Composable

fun Calculator3App(modifier: Modifier = Modifier) {

val focusManager = *LocalFocusManager*.current

// визначаю змінні

var S\_b by remember **{** *mutableStateOf*("1000") **}**

var U\_b by remember **{** *mutableStateOf*("6.3") **}**

var S\_k by remember **{** *mutableStateOf*("200") **}**

var U\_k\_perc by remember **{** *mutableStateOf*("7.5") **}**

var S\_nom\_t by remember **{** *mutableStateOf*("4") **}**

var X0 by remember **{** *mutableStateOf*("0.08") **}**

var l by remember **{** *mutableStateOf*("0.3") **}**

var X\_d\_perc by remember **{** *mutableStateOf*("23.8") **}**

var S\_nom\_d by remember **{** *mutableStateOf*("1.94") **}**

var Ec by remember **{** *mutableStateOf*("1") **}**

var Ed by remember **{** *mutableStateOf*("1.07") **}**

var t by *remember* **{** *mutableStateOf*("0.065") **}**

var Yt by *remember* **{** *mutableStateOf*("0.71") **}**

var Tac by *remember* **{** *mutableStateOf*("0.03") **}**

var Tad by *remember* **{** *mutableStateOf*("0.037") **}**

var Tpz by *remember* **{** *mutableStateOf*("0.5") **}**

var Tpv by *remember* **{** *mutableStateOf*("0.1") **}**

var Ib by *remember* **{** *mutableStateOf*("") **}**

var Xc by *remember* **{** *mutableStateOf*("") **}**

var Xt by *remember* **{** *mutableStateOf*("") **}**

var Xl by *remember* **{** *mutableStateOf*("") **}**

var Xd by *remember* **{** *mutableStateOf*("") **}**

var U\_c\_n by *remember* **{** *mutableStateOf*("") **}**

var X\_E\_c by *remember* **{** *mutableStateOf*("") **}**

var X\_E\_d by *remember* **{** *mutableStateOf*("") **}**

var Ip0c by *remember* **{** *mutableStateOf*("") **}**

var Ip0d by *remember* **{** *mutableStateOf*("") **}**

var Iptc by *remember* **{** *mutableStateOf*("") **}**

var Iptd by *remember* **{** *mutableStateOf*("") **}**

var Iac by *remember* **{** *mutableStateOf*("") **}**

var Iad by *remember* **{** *mutableStateOf*("") **}**

var Iuds by *remember* **{** *mutableStateOf*("") **}**

var Iudd by *remember* **{** *mutableStateOf*("") **}**

var Tach by *remember* **{** *mutableStateOf*("") **}**

var Tpd by *remember* **{** *mutableStateOf*("") **}**

var Tvidk by *remember* **{** *mutableStateOf*("") **}**

var Vk by *remember* **{** *mutableStateOf*("") **}**

// розмітка застосунку

*Column*(

modifier = Modifier

.*fillMaxSize*()

.*padding*(16.*dp*)

.*verticalScroll*(*rememberScrollState*())

) **{**

*Spacer*(modifier = Modifier.*height*(8.*dp*))

*OutlinedTextField*(

value = S\_b,

onValueChange = **{** S\_b = **it }**,

label = **{** *Text*("S\_b") **}**,

modifier = Modifier

.*fillMaxWidth*()

.*clickable* **{** focusManager.clearFocus() **}**

)

*Spacer*(modifier = Modifier.*height*(8.*dp*))

*OutlinedTextField*(

value = U\_b,

onValueChange = **{** U\_b = **it }**,

label = **{** *Text*("U\_b") **}**,

modifier = Modifier

.*fillMaxWidth*()

.*clickable* **{** focusManager.clearFocus() **}**

)

*Spacer*(modifier = Modifier.*height*(8.*dp*))

*OutlinedTextField*(

value = S\_k,

onValueChange = **{** S\_k = **it }**,

label = **{** *Text*("S\_k") **}**,

modifier = Modifier

.*fillMaxWidth*()

.*clickable* **{** focusManager.clearFocus() **}**

)

*Spacer*(modifier = Modifier.*height*(8.*dp*))

*OutlinedTextField*(

value = U\_k\_perc,

onValueChange = **{** U\_k\_perc = **it }**,

label = **{** *Text*("U\_k\_perc") **}**,

modifier = Modifier

.*fillMaxWidth*()

.*clickable* **{** focusManager.clearFocus() **}**

)

*Spacer*(modifier = Modifier.*height*(8.*dp*))

*OutlinedTextField*(

value = S\_nom\_t,

onValueChange = **{** S\_nom\_t = **it }**,

label = **{** *Text*("S\_nom\_t") **}**,

modifier = Modifier

.*fillMaxWidth*()

.*clickable* **{** focusManager.clearFocus() **}**

)

*Spacer*(modifier = Modifier.*height*(8.*dp*))

*OutlinedTextField*(

value = X0,

onValueChange = **{** X0 = **it }**,

label = **{** *Text*("X0") **}**,

modifier = Modifier

.*fillMaxWidth*()

.*clickable* **{** focusManager.clearFocus() **}**

)

*Spacer*(modifier = Modifier.*height*(8.*dp*))

*OutlinedTextField*(

value = l,

onValueChange = **{** l = **it }**,

label = **{** *Text*("l") **}**,

modifier = Modifier

.*fillMaxWidth*()

.*clickable* **{** focusManager.clearFocus() **}**

)

*Spacer*(modifier = Modifier.*height*(8.*dp*))

*OutlinedTextField*(

value = X\_d\_perc,

onValueChange = **{** X\_d\_perc = **it }**,

label = **{** *Text*("X\_d\_perc") **}**,

modifier = Modifier

.*fillMaxWidth*()

.*clickable* **{** focusManager.clearFocus() **}**

)

*Spacer*(modifier = Modifier.*height*(8.*dp*))

*OutlinedTextField*(

value = S\_nom\_d,

onValueChange = **{** S\_nom\_d = **it }**,

label = **{** *Text*("S\_nom\_d") **}**,

modifier = Modifier

.*fillMaxWidth*()

.*clickable* **{** focusManager.clearFocus() **}**

)

*Spacer*(modifier = Modifier.*height*(8.*dp*))

*OutlinedTextField*(

value = Ec,

onValueChange = **{** Ec = **it }**,

label = **{** *Text*("Ec") **}**,

modifier = Modifier

.*fillMaxWidth*()

.*clickable* **{** focusManager.clearFocus() **}**

)

*Spacer*(modifier = Modifier.*height*(8.*dp*))

*OutlinedTextField*(

value = Ed,

onValueChange = **{** Ed = **it }**,

label = **{** *Text*("Ed") **}**,

modifier = Modifier

.*fillMaxWidth*()

.*clickable* **{** focusManager.clearFocus() **}**

)

*Spacer*(modifier = Modifier.*height*(8.*dp*))

*OutlinedTextField*(

value = t,

onValueChange = **{** t = **it }**,

label = **{** *Text*("t") **}**,

modifier = Modifier

.*fillMaxWidth*()

.*clickable* **{** focusManager.clearFocus() **}**

)

*Spacer*(modifier = Modifier.*height*(8.*dp*))

*OutlinedTextField*(

value = Yt,

onValueChange = **{** Yt = **it }**,

label = **{** *Text*("Yt") **}**,

modifier = Modifier

.*fillMaxWidth*()

.*clickable* **{** focusManager.clearFocus() **}**

)

*Spacer*(modifier = Modifier.*height*(8.*dp*))

*OutlinedTextField*(

value = Tac,

onValueChange = **{** Tac = **it }**,

label = **{** *Text*("Tac") **}**,

modifier = Modifier

.*fillMaxWidth*()

.*clickable* **{** focusManager.clearFocus() **}**

)

*Spacer*(modifier = Modifier.*height*(8.*dp*))

*OutlinedTextField*(

value = Tad,

onValueChange = **{** Tad = **it }**,

label = **{** *Text*("Tad") **}**,

modifier = Modifier

.*fillMaxWidth*()

.*clickable* **{** focusManager.clearFocus() **}**

)

*Spacer*(modifier = Modifier.*height*(8.*dp*))

*OutlinedTextField*(

value = Tpz,

onValueChange = **{** Tpz = **it }**,

label = **{** *Text*("Tpz") **}**,

modifier = Modifier

.*fillMaxWidth*()

.*clickable* **{** focusManager.clearFocus() **}**

)

*Spacer*(modifier = Modifier.*height*(8.*dp*))

*OutlinedTextField*(

value = Tpv,

onValueChange = **{** Tpv = **it }**,

label = **{** *Text*("Tpv") **}**,

modifier = Modifier

.*fillMaxWidth*()

.*clickable* **{** focusManager.clearFocus() **}**

)

*Spacer*(modifier = Modifier.*height*(8.*dp*))

// кнопка для виконання обчислень

*Button*(

onClick = **{**

val \_S\_b = S\_b.*toDoubleOrNull*() ?: 0.0

val \_U\_b = U\_b.*toDoubleOrNull*() ?: 0.0

val \_S\_k = S\_k.*toDoubleOrNull*() ?: 0.0

val \_U\_k\_perc = U\_k\_perc.*toDoubleOrNull*() ?: 0.0

val \_S\_nom\_t = S\_nom\_t.*toDoubleOrNull*() ?: 0.0

val \_X0 = X0.*toDoubleOrNull*() ?: 0.0

val \_l = l.*toDoubleOrNull*() ?: 0.0

val \_X\_d\_perc = X\_d\_perc.*toDoubleOrNull*() ?: 0.0

val \_S\_nom\_d = S\_nom\_d.*toDoubleOrNull*() ?: 0.0

val \_Ec = Ec.*toDoubleOrNull*() ?: 0.0

val \_Ed = Ed.*toDoubleOrNull*() ?: 0.0

val \_t = t.*toDoubleOrNull*() ?: 0.0

val \_Yt = Yt.*toDoubleOrNull*() ?: 0.0

val \_Tac = Tac.*toDoubleOrNull*() ?: 0.0

val \_Tad = Tad.*toDoubleOrNull*() ?: 0.0

val \_Tpz = Tpz.*toDoubleOrNull*() ?: 0.0

val \_Tpv = Tpv.*toDoubleOrNull*() ?: 0.0

var \_Ib = Ib.*toDoubleOrNull*() ?: 0.0

var \_Xc = Xc.*toDoubleOrNull*() ?: 0.0

var \_Xt = Xt.*toDoubleOrNull*() ?: 0.0

var \_Xl = Xl.*toDoubleOrNull*() ?: 0.0

var \_Xd = Xd.*toDoubleOrNull*() ?: 0.0

var \_U\_c\_n = U\_c\_n.*toDoubleOrNull*() ?: 0.0

var \_X\_E\_c = X\_E\_c.*toDoubleOrNull*() ?: 0.0

var \_X\_E\_d = X\_E\_d.*toDoubleOrNull*() ?: 0.0

var \_Ip0c = Ip0c.*toDoubleOrNull*() ?: 0.0

var \_Ip0d = Ip0d.*toDoubleOrNull*() ?: 0.0

var \_Iptc = Iptc.*toDoubleOrNull*() ?: 0.0

var \_Iptd = Iptd.*toDoubleOrNull*() ?: 0.0

var \_Iac = Iac.*toDoubleOrNull*() ?: 0.0

var \_Iad = Iad.*toDoubleOrNull*() ?: 0.0

var \_Iuds = Iuds.*toDoubleOrNull*() ?: 0.0

var \_Iudd = Iudd.*toDoubleOrNull*() ?: 0.0

var \_Tach = Tach.*toDoubleOrNull*() ?: 0.0

var \_Tpd = Tpd.*toDoubleOrNull*() ?: 0.0

var \_Tvidk = Tvidk.*toDoubleOrNull*() ?: 0.0

var \_Vk = Vk.*toDoubleOrNull*() ?: 0.0

\_Ib = \_S\_b / (*sqrt*(3.0) \* \_U\_b)

\_Xc = \_S\_b / \_S\_k

\_Xt = \_U\_k\_perc / 100 \* \_S\_b / \_S\_nom\_t

\_U\_c\_n = \_U\_b \* \_U\_b

\_Xl = \_X0 \* \_l \* \_S\_b / \_U\_c\_n

\_Xd = \_X\_d\_perc / 100 \* \_S\_b / \_S\_nom\_d

\_X\_E\_c = \_Xc + \_Xt

\_X\_E\_d = \_Xd + \_Xl

\_Ip0c = \_Ec / \_X\_E\_c \* \_Ib

\_Ip0d = \_Ed / \_X\_E\_d \* \_Ib

\_Iptc = \_Ip0c

\_Iptd = \_Ip0d \* \_Yt

\_Iac = *sqrt*(2.0) \* \_Ip0c \* *exp*(-\_t / \_Tac)

\_Iad = *sqrt*(2.0) \* \_Ip0d \* *exp*(-\_t / \_Tad)

\_Iuds = *sqrt*(2.0) \* \_Ip0c \* (1 + *exp*(-0.01 / \_Tac))

\_Iudd = *sqrt*(2.0) \* \_Ip0d \* (1 + *exp*(-0.01 / \_Tad))

\_Tach = (\_Tac \* \_Ip0c + \_Tad \* \_Ip0d) / (\_Ip0c + \_Ip0d)

\_Tpd = -\_t / (*ln*(\_t))

\_Tvidk = \_Tpz + \_Tpv

\_Vk = (\_Ip0c \* \_Ip0c \* (\_Tvidk + \_Tach)) + (\_Ip0d \* \_Ip0d \* (0.5 \* \_Tpd + \_Tach)) + (2 \* \_Ip0c \* \_Ip0d \* (\_Tpd + \_Tach))

Log.d("CalculationResults", "\_Ib = $\_Ib")

Log.d("CalculationResults", "\_Xc = $\_Xc")

Log.d("CalculationResults", "\_Xt = $\_Xt")

Log.d("CalculationResults", "\_U\_c\_n = $\_U\_c\_n")

Log.d("CalculationResults", "\_Xl = $\_Xl")

Log.d("CalculationResults", "\_Xd = $\_Xd")

Log.d("CalculationResults", "\_X\_E\_c = $\_X\_E\_c")

Log.d("CalculationResults", "\_X\_E\_d = $\_X\_E\_d")

Log.d("CalculationResults", "\_Ip0c = $\_Ip0c")

Log.d("CalculationResults", "\_Ip0d = $\_Ip0d")

Log.d("CalculationResults", "\_Iptc = $\_Iptc")

Log.d("CalculationResults", "\_Iptd = $\_Iptd")

Log.d("CalculationResults", "\_Iac = $\_Iac")

Log.d("CalculationResults", "\_Iad = $\_Iad")

Log.d("CalculationResults", "\_Iuds = $\_Iuds")

Log.d("CalculationResults", "\_Iudd = $\_Iudd")

Log.d("CalculationResults", "\_Tach = $\_Tach")

Log.d("CalculationResults", "\_Tpd = $\_Tpd")

Log.d("CalculationResults", "\_Tvidk = $\_Tvidk")

Log.d("CalculationResults", "\_Vk = $\_Vk")

Ib = *roundToTwoDecimalString*(\_Ib)

Xc = *roundToTwoDecimalString*(\_Xc)

Xt = *roundToTwoDecimalString*(\_Xt)

U\_c\_n = *roundToTwoDecimalString*(\_U\_c\_n)

Xl = *roundToTwoDecimalString*(\_Xl)

Xd = *roundToTwoDecimalString*(\_Xd)

X\_E\_c = *roundToTwoDecimalString*(\_X\_E\_c)

X\_E\_d = *roundToTwoDecimalString*(\_X\_E\_d)

Ip0c = *roundToTwoDecimalString*(\_Ip0c)

Ip0d = *roundToTwoDecimalString*(\_Ip0d)

Iptc = *roundToTwoDecimalString*(\_Iptc)

Iptd = *roundToTwoDecimalString*(\_Iptd)

Iac = *roundToTwoDecimalString*(\_Iac)

Iad = *roundToTwoDecimalString*(\_Iad)

Iuds = *roundToTwoDecimalString*(\_Iuds)

Iudd = *roundToTwoDecimalString*(\_Iudd)

Tach = *roundToTwoDecimalString*(\_Tach)

Tpd = *roundToTwoDecimalString*(\_Tpd)

Tvidk = *roundToTwoDecimalString*(\_Tvidk)

Vk = *roundToTwoDecimalString*(\_Vk)

**}**,

modifier = Modifier.*fillMaxWidth*()

) **{**

*Text*("Обчислити")

**}**

// вивід результатів

*Text*(text = "Ib: " + Ib, style = MaterialTheme.typography.bodyLarge)

*Spacer*(modifier = Modifier.*height*(8.*dp*))

*Text*(text = "Xc: " + Xc, style = MaterialTheme.typography.bodyLarge)

*Spacer*(modifier = Modifier.*height*(8.*dp*))

*Text*(text = "Xt: " + Xt, style = MaterialTheme.typography.bodyLarge)

*Spacer*(modifier = Modifier.*height*(8.*dp*))

*Text*(text = "U\_c\_n: " + U\_c\_n, style = MaterialTheme.typography.bodyLarge)

*Spacer*(modifier = Modifier.*height*(8.*dp*))

*Text*(text = "Xl: " + Xl, style = MaterialTheme.typography.bodyLarge)

*Spacer*(modifier = Modifier.*height*(8.*dp*))

*Text*(text = "Xd: " + Xd, style = MaterialTheme.typography.bodyLarge)

*Spacer*(modifier = Modifier.*height*(8.*dp*))

*Text*(text = "X\_E\_c: " + X\_E\_c, style = MaterialTheme.typography.bodyLarge)

*Spacer*(modifier = Modifier.*height*(8.*dp*))

*Text*(text = "X\_E\_d: " + X\_E\_d, style = MaterialTheme.typography.bodyLarge)

*Spacer*(modifier = Modifier.*height*(8.*dp*))

*Text*(text = "Ip0c: " + Ip0c, style = MaterialTheme.typography.bodyLarge)

*Spacer*(modifier = Modifier.*height*(8.*dp*))

*Text*(text = "Ip0d: " + Ip0d, style = MaterialTheme.typography.bodyLarge)

*Spacer*(modifier = Modifier.*height*(8.*dp*))

*Text*(text = "Iptc: " + Iptc, style = MaterialTheme.typography.bodyLarge)

*Spacer*(modifier = Modifier.*height*(8.*dp*))

*Text*(text = "Iptd: " + Iptd, style = MaterialTheme.typography.bodyLarge)

*Spacer*(modifier = Modifier.*height*(8.*dp*))

*Text*(text = "Iac: " + Iac, style = MaterialTheme.typography.bodyLarge)

*Spacer*(modifier = Modifier.*height*(8.*dp*))

*Text*(text = "Iad: " + Iad, style = MaterialTheme.typography.bodyLarge)

*Spacer*(modifier = Modifier.*height*(8.*dp*))

*Text*(text = "Iuds: " + Iuds, style = MaterialTheme.typography.bodyLarge)

*Spacer*(modifier = Modifier.*height*(8.*dp*))

*Text*(text = "Iudd: " + Iudd, style = MaterialTheme.typography.bodyLarge)

*Spacer*(modifier = Modifier.*height*(8.*dp*))

*Text*(text = "Tach: " + Tach, style = MaterialTheme.typography.bodyLarge)

*Spacer*(modifier = Modifier.*height*(8.*dp*))

*Text*(text = "Tpd: " + Tpd, style = MaterialTheme.typography.bodyLarge)

*Spacer*(modifier = Modifier.*height*(8.*dp*))

*Text*(text = "Tvidk: " + Tvidk, style = MaterialTheme.typography.bodyLarge)

*Spacer*(modifier = Modifier.*height*(8.*dp*))

*Text*(text = "Vk: " + Vk, style = MaterialTheme.typography.bodyLarge)

*Spacer*(modifier = Modifier.*height*(8.*dp*))

**}**

}

fun roundToTwoDecimalString(value: Double): String {

return (Math.round(value \* 100) / 100.0).toString()

}

А також додамо стилів:

styles.xml:  
<resources>

<style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">

<item name="colorPrimary">#6200EE</item>

<item name="colorPrimaryDark">#3700B3</item>

<item name="colorAccent">#03DAC5</item>

</style>

</resources>

themes.xml:  
<?xml version="1.0" encoding="utf-8"?>

<resources>

<style name="Theme.Kpikotlinlab4" parent="Theme.AppCompat.Light.NoActionBar">

<!-- Основні кольори теми -->

<item name="colorPrimary">#6200EE</item>

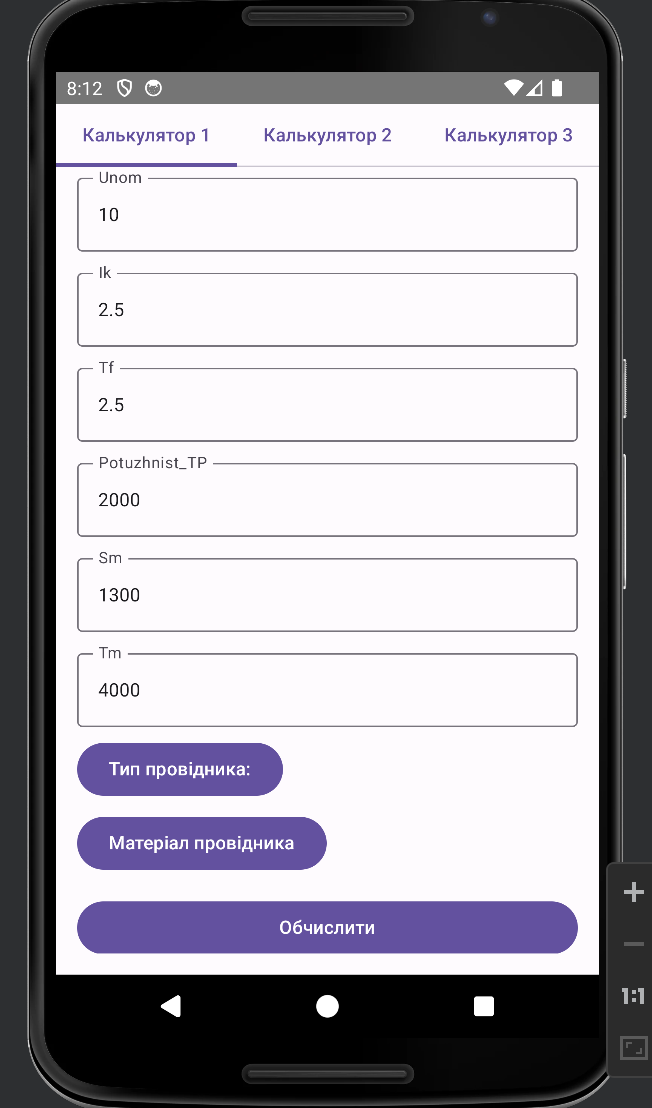
<item name="colorPrimaryDark">#3700B3</item>

<item name="colorAccent">#03DAC5</item>

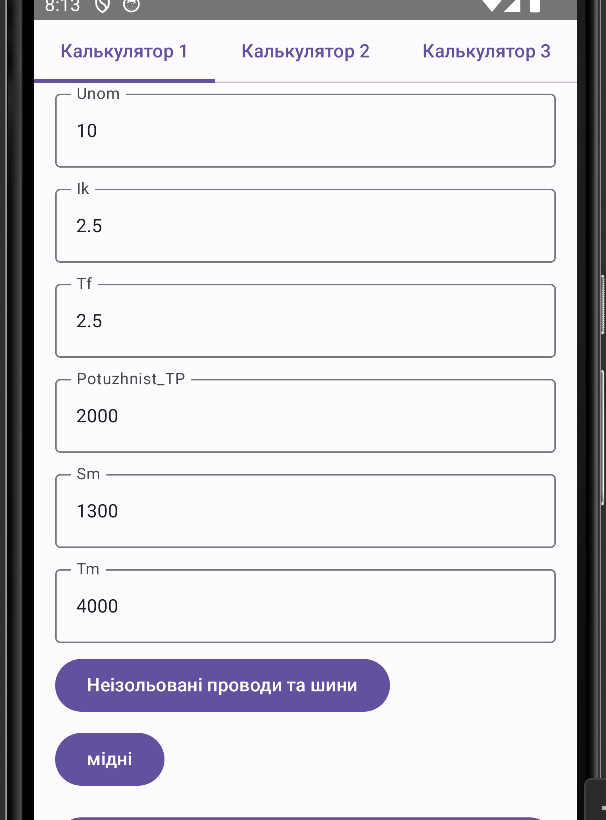
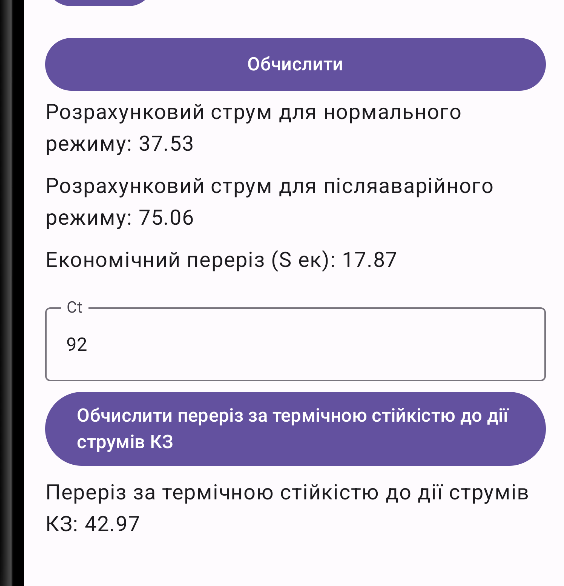
</style>

</resources>

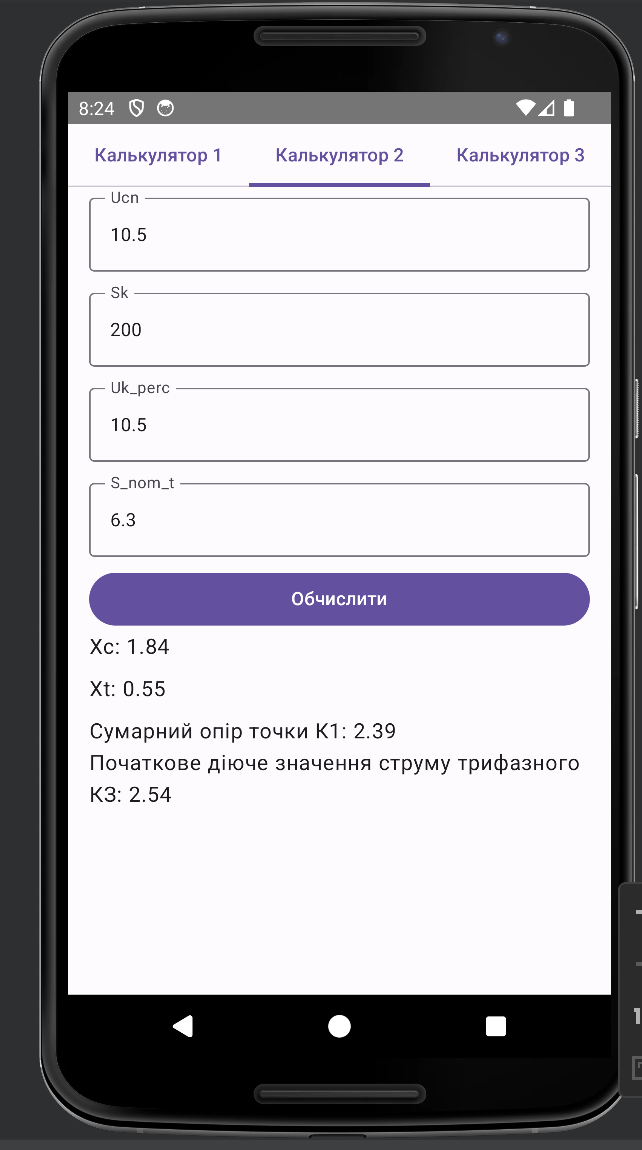
Головний екран виглядатиме наступним чином:



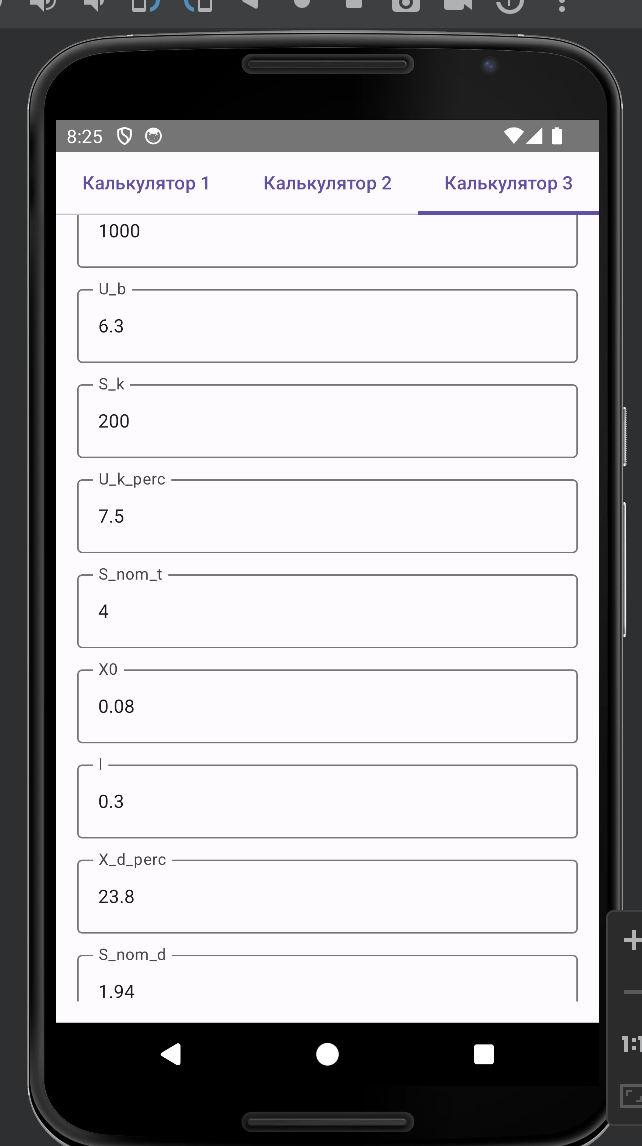
Результат виконання (**Завдання 1**):

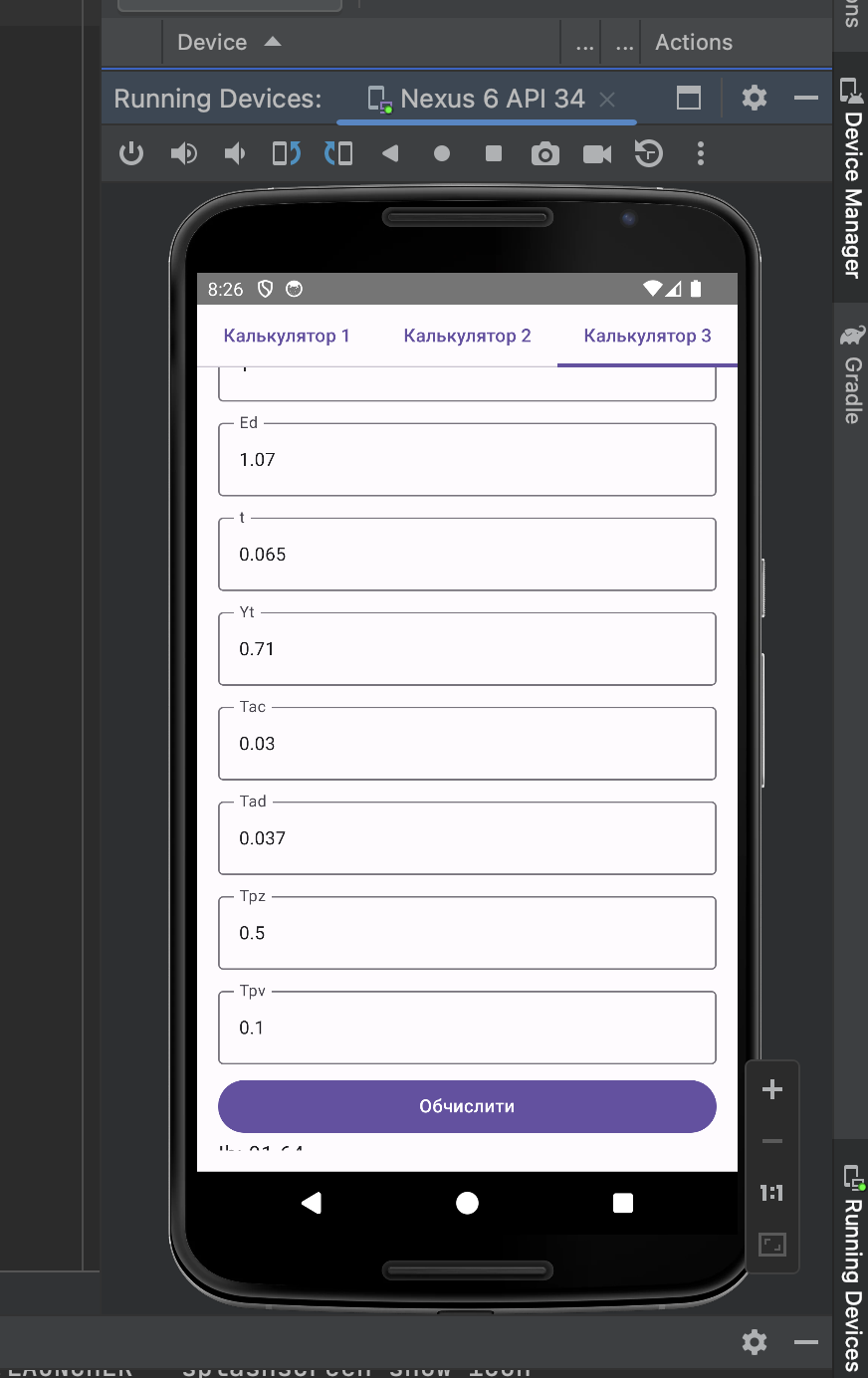


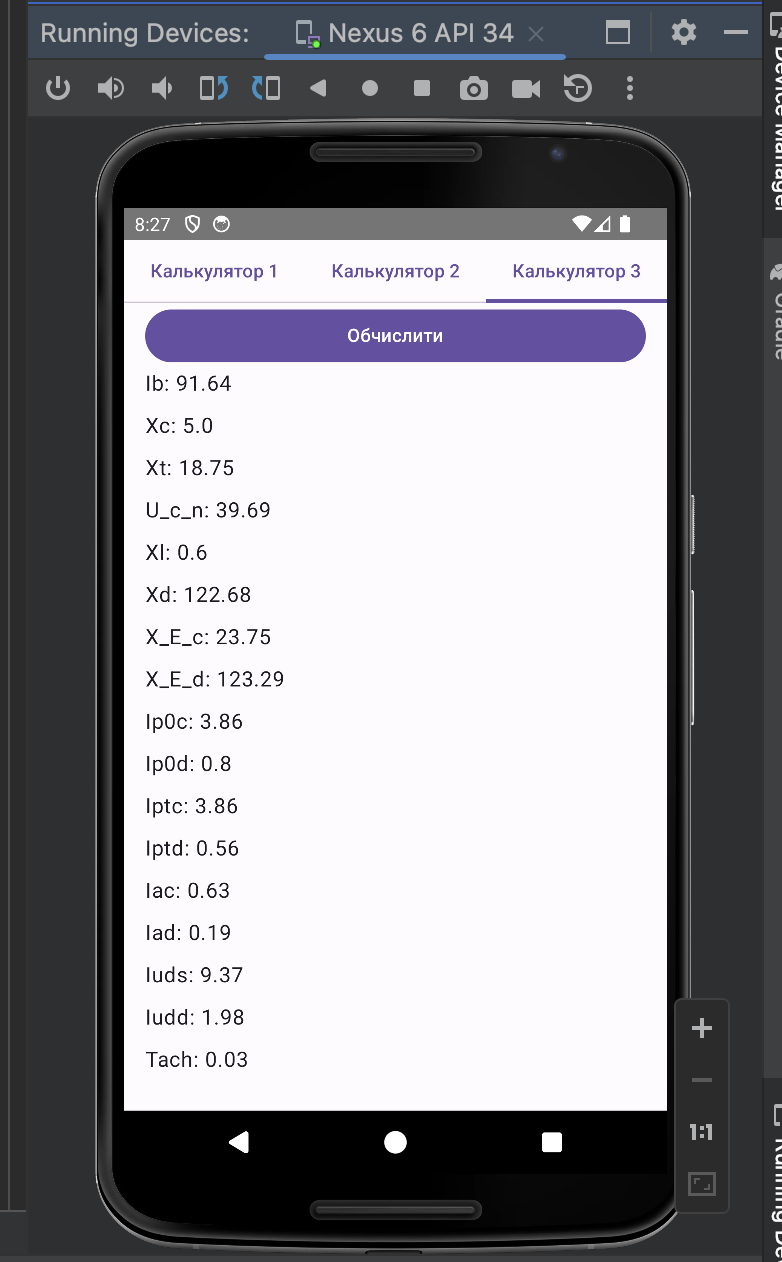
Результат виконання (**Завдання 2**):



Результат виконання (**Завдання 3**):







**Висновок**

У процесі виконання лабораторної роботи було розроблено мобільний додаток для платформи Android, який дозволяє виконувати розрахунки струмів трифазного та однофазного короткого замикання, а також перевіряти кабелі на термічну та динамічну стійкість. Основною метою було підібрати відповідні кабелі для двотрансформаторної підстанції, розрахувати струми короткого замикання на шинах 10 кВ головної понижувальної підстанції (ГПП) та для підстанції Хмельницьких північних електричних мереж (ХПнЕМ) у різних режимах роботи: нормальному, мінімальному та аварійному.

Для створення інтерфейсу користувача було використано мову програмування Kotlin із застосуванням таких компонентів, як DropdownMenu, таби для поділу функціоналу калькуляторів, а також структур listOf та mapOf для організації даних. Для виконання обчислень використовувалася бібліотека math, а також інші елементи Jetpack Compose. Виконані розрахунки підтвердили правильність використаних формул та методик, відповідно до прикладів, наведених у завданні.

Ця лабораторна робота дозволила отримати практичні навички у створенні Android-додатків для електротехнічних розрахунків, а також у використанні Kotlin для розробки інтерактивного інтерфейсу та реалізації математичних обчислень.