Міністерство освіти і науки України  
Національний технічний університет України

«Київський політехнічний інститут ім. І. Сікорського»

Кафедра інженерії програмного забезпечення в енергетиці

Лабораторна робота № 1

з курсу: «*Кросплатформна розробка мобільних застосунків*»

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

Посилання на GitHub репозиторій:

<https://github.com/Chivas1717/cross-platform-labs/blob/master/app/lib/calculators/lab1.dart>

**Перевірив:**

Недашківський О.Л.

Київ 2025

Лабораторна робота № 1

Варіант №4

**Хід виконання:**

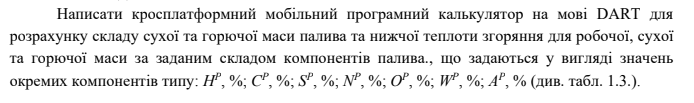
Метою роботи є створення кросплатформного мобільного застосунку для розрахунку складу сухої та горючої маси палива, а також нижчої теплоти згоряння. Додаток реалізований на мові Dart з використанням фреймворку Flutter та підтримує роботу на мобільних і десктопних пристроях.

### **2. Опис програмної реалізації**

Для реалізації було створено два калькулятори:

* **Завдання 1**: Розрахунок складу сухої та горючої маси палива.
* **Завдання 2**: Розрахунок параметрів мазуту, включаючи нижчу теплоту згоряння.

У програмі використовується ToggleButtons для перемикання між завданнями, а розрахунки виконуються відповідно до наведених у методичці формул.

**Завдання 1:**

class Task1Calculator extends StatefulWidget {

@override

\_Task1CalculatorState createState() => \_Task1CalculatorState();

}

class \_Task1CalculatorState extends State<Task1Calculator> {

// Контролери для текстових полів

final Map<String, TextEditingController> controllers = {

'H': TextEditingController(text: '7.0'), // Варіант №4

'C': TextEditingController(text: '60.0'),

'S': TextEditingController(text: '2.5'),

'N': TextEditingController(text: '0.5'),

'O': TextEditingController(text: '10.5'),

'W': TextEditingController(text: '10.5'),

'A': TextEditingController(text: '9.5'),

};

double? dryMass; // krs (від робочої до сухої маси)

double? combustibleMass; // krg (від робочої до горючої маси)

double? qph; // Нижча теплота згоряння робочої маси

double? qdh; // Нижча теплота згоряння сухої маси

double? qdafh; // Нижча теплота згоряння горючої маси

double? hDry, cDry, sDry, nDry, oDry, aDry;

double? hComb, cComb, sComb, nComb, oComb, aComb;

void calculateTask1() {

setState(() {

double h = double.*tryParse*(controllers['H']!.text) ?? 0;

double c = double.*tryParse*(controllers['C']!.text) ?? 0;

double s = double.*tryParse*(controllers['S']!.text) ?? 0;

double n = double.*tryParse*(controllers['N']!.text) ?? 0;

double o = double.*tryParse*(controllers['O']!.text) ?? 0;

double w = double.*tryParse*(controllers['W']!.text) ?? 0;

double a = double.*tryParse*(controllers['A']!.text) ?? 0;

// Коефіцієнти переходу

double krs = 100 / (100 - w); // до сухої маси

double krg = 100 / (100 - w - a); // до горючої маси

dryMass = krs;

combustibleMass = krg;

// Нижча теплота згоряння робочої маси

qph = (339 \* c + 1030 \* h - 108.8 \* (o - s) - 25 \* w) / 1000;

qdh = qph! \* krs;

qdafh = qph! \* krg;

// Склад сухої маси

hDry = h \* krs;

cDry = c \* krs;

sDry = s \* krs;

nDry = n \* krs;

oDry = o \* krs;

aDry = a \* krs;

// Склад горючої маси

hComb = h \* krg;

cComb = c \* krg;

sComb = s \* krg;

nComb = n \* krg;

oComb = o \* krg;

aComb = a \* krg;

});

}

@override

Widget build(BuildContext context) {

String inputComposition = 'H=${controllers['H']!.text}%; '

'C=${controllers['C']!.text}%; '

'S=${controllers['S']!.text}%; '

'N=${controllers['N']!.text}%; '

'O=${controllers['O']!.text}%; '

'W=${controllers['W']!.text}%; '

'A=${controllers['A']!.text}%';

return SingleChildScrollView(

padding: EdgeInsets.all(16.0),

child: Column(

crossAxisAlignment: CrossAxisAlignment.start,

children: [

...controllers.entries.map((entry) {

return Padding(

padding: const EdgeInsets.symmetric(vertical: 6.0),

child: TextField(

controller: entry.value,

decoration: InputDecoration(

labelText: '${entry.key}, %',

border: OutlineInputBorder(),

),

keyboardType: TextInputType.*number*,

),

);

}).toList(),

SizedBox(height: 16.0),

ElevatedButton(

onPressed: calculateTask1,

child: Text('Calculate'),

),

SizedBox(height: 24.0),

if (qph != null) ...[

Text(

'1. Для палива з компонентним складом: $inputComposition',

style: TextStyle(fontWeight: FontWeight.*bold*),

),

SizedBox(height: 8),

Text(

'1.1. Коефіцієнт переходу від робочої до сухої маси становить: '

'${dryMass?.toStringAsFixed(3)}',

),

Text(

'1.2. Коефіцієнт переходу від робочої до горючої маси становить: '

'${combustibleMass?.toStringAsFixed(3)}',

),

SizedBox(height: 8),

Text(

'1.3. Склад сухої маси палива становитиме:\n'

' Hс=${hDry?.toStringAsFixed(2)}%; '

'Cс=${cDry?.toStringAsFixed(2)}%; '

'Sс=${sDry?.toStringAsFixed(2)}%; '

'Nс=${nDry?.toStringAsFixed(2)}%; '

'Oс=${oDry?.toStringAsFixed(2)}%; '

'Aс=${aDry?.toStringAsFixed(2)}%',

),

SizedBox(height: 8),

Text(

'1.4. Склад горючої маси палива становитиме:\n'

' Hг=${hComb?.toStringAsFixed(2)}%; '

'Cг=${cComb?.toStringAsFixed(2)}%; '

'Sг=${sComb?.toStringAsFixed(2)}%; '

'Nг=${nComb?.toStringAsFixed(2)}%; '

'Oг=${oComb?.toStringAsFixed(2)}%; '

'Aг=${aComb?.toStringAsFixed(2)}%',

),

SizedBox(height: 8),

Text(

'1.5. Нижча теплота згоряння для робочої маси: '

'${qph?.toStringAsFixed(4)} МДж/кг',

),

Text(

'1.6. Нижча теплота згоряння для сухої маси: '

'${qdh?.toStringAsFixed(4)} МДж/кг',

),

Text(

'1.7. Нижча теплота згоряння для горючої маси: '

'${qdafh?.toStringAsFixed(4)} МДж/кг',

),

SizedBox(height: 32),

],

],

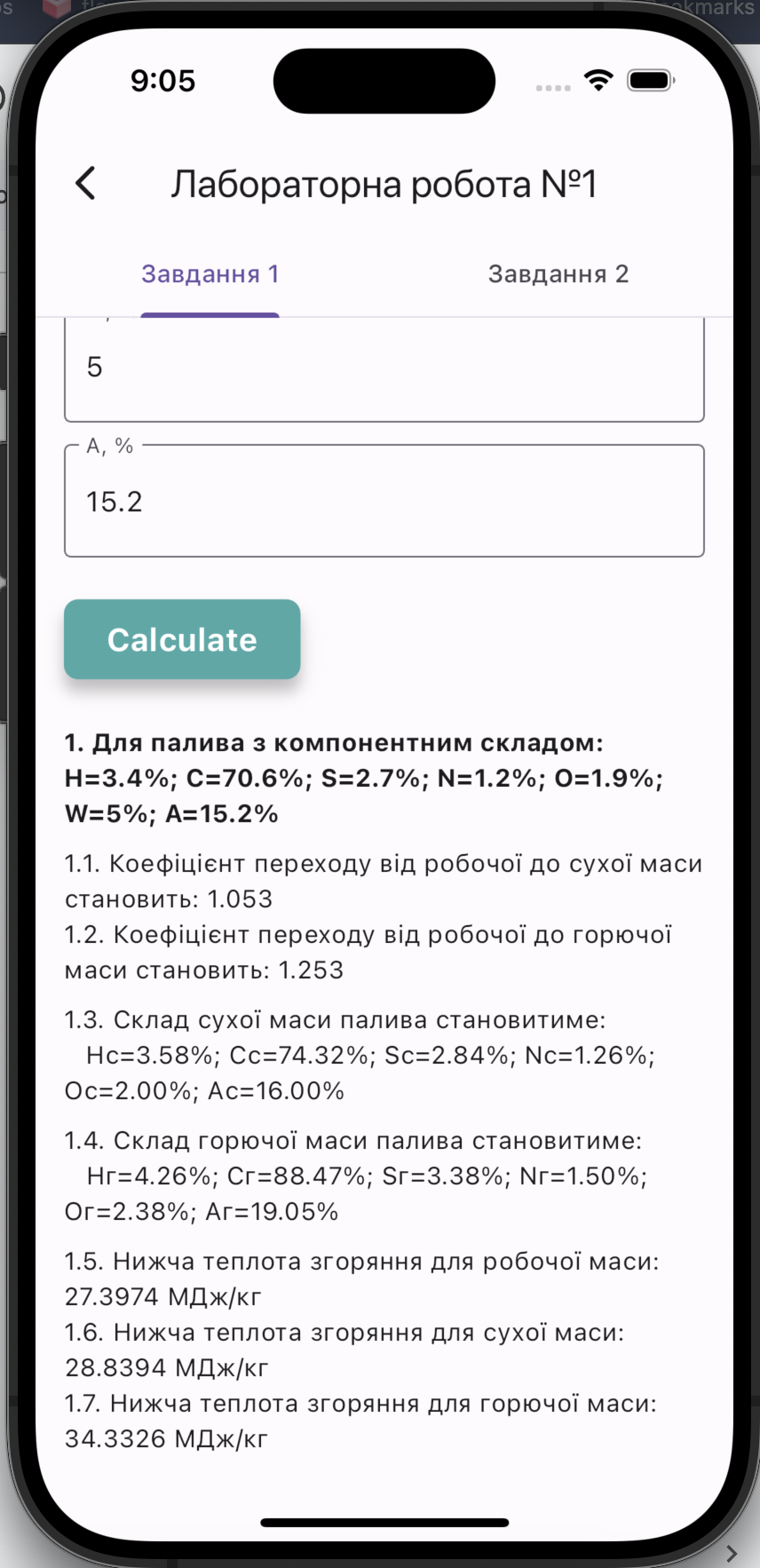
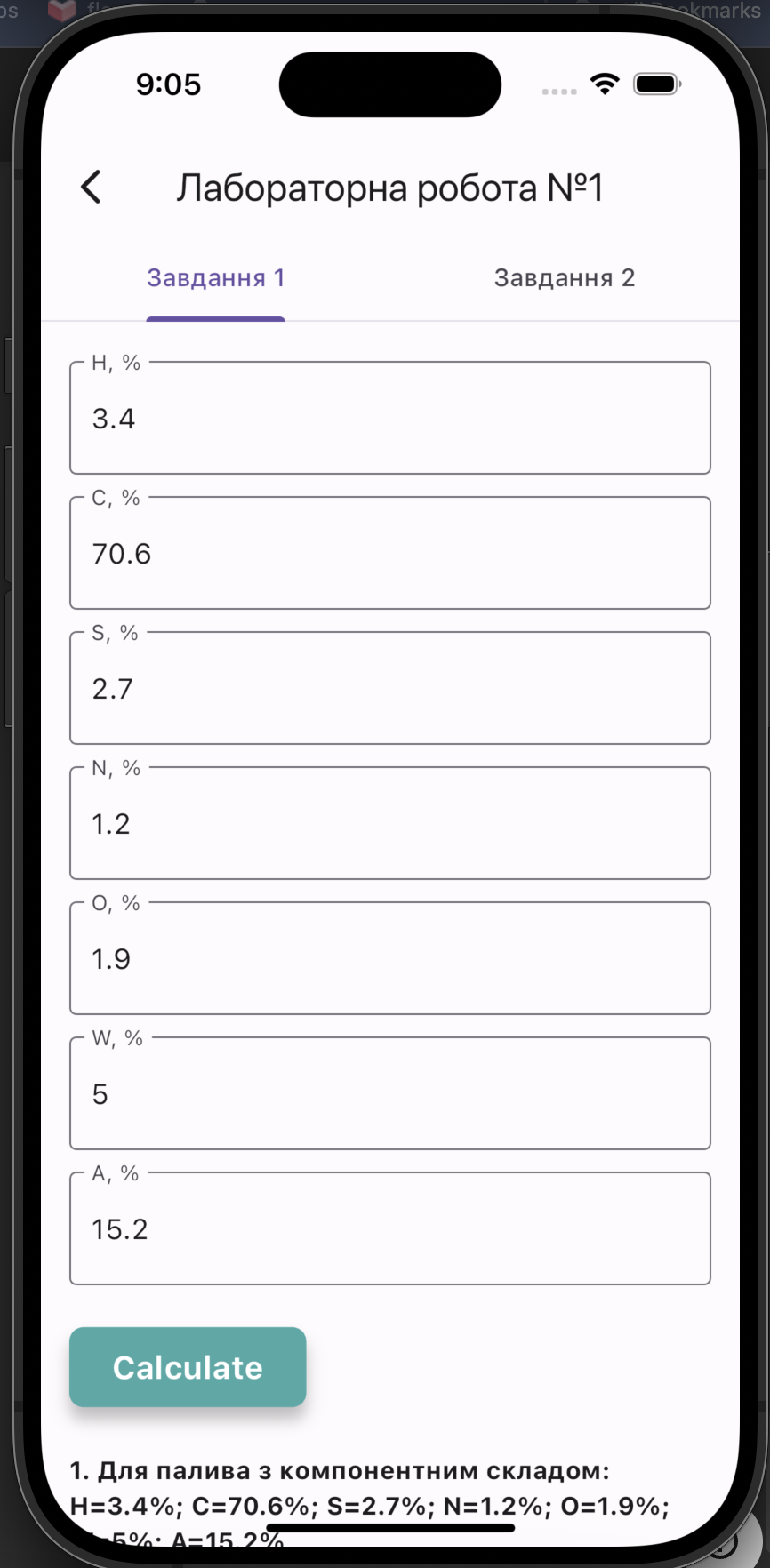
),

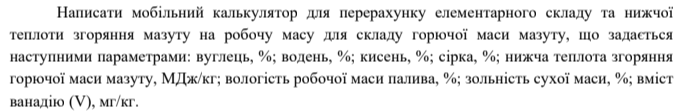
);

}

}

Результат перевірки для варіанта №4:



**Завдання 2:** ****

class Task2Calculator extends StatefulWidget {

@override

\_Task2CalculatorState createState() => \_Task2CalculatorState();

}

class \_Task2CalculatorState extends State<Task2Calculator> {

final Map<String, TextEditingController> controllers = {

// Контрольний приклад «горючої» маси мазуту:

'C': TextEditingController(text: '85.50'),

'H': TextEditingController(text: '11.20'),

'S': TextEditingController(text: '2.50'),

'O': TextEditingController(text: '0.80'),

'V': TextEditingController(text: '333.3'), // mg/kg

'W': TextEditingController(text: '2.00'),

'A': TextEditingController(text: '0.15'),

'Qdaf': TextEditingController(text: '40.40'), // МДж/кг

};

double? qWork; // нижча теплота згоряння для робочої маси

// Щоби показати «склад робочої маси»:

double? cWork, hWork, sWork, oWork, vWork;

double? wWork, aWork;

void calculateTask2() {

setState(() {

double c = double.*tryParse*(controllers['C']!.text) ?? 0;

double h = double.*tryParse*(controllers['H']!.text) ?? 0;

double s = double.*tryParse*(controllers['S']!.text) ?? 0;

double o = double.*tryParse*(controllers['O']!.text) ?? 0;

double v = double.*tryParse*(controllers['V']!.text) ?? 0; // mg/kg

double w = double.*tryParse*(controllers['W']!.text) ?? 0;

double a = double.*tryParse*(controllers['A']!.text) ?? 0;

double qDaf = double.*tryParse*(controllers['Qdaf']!.text) ?? 0;

// Формула для нижчої теплоти згоряння на робочу масу

qWork = qDaf \* (100 - w - a) / 100 - 0.025 \* w;

// Перехід від «горючої» маси до «робочої»:

double ratio = (100 - w - a) / 100;

cWork = (c / 100) \* (100 - w - a);

hWork = (h / 100) \* (100 - w - a);

sWork = (s / 100) \* (100 - w - a);

oWork = (o / 100) \* (100 - w - a);

vWork = v \* ratio;

wWork = w;

aWork = a;

});

}

@override

Widget build(BuildContext context) {

String inputComposition = 'Hг=${controllers['H']!.text}%; '

'Cг=${controllers['C']!.text}%; '

'Sг=${controllers['S']!.text}%; '

'Oг=${controllers['O']!.text}%; '

'Vг=${controllers['V']!.text} мг/кг; '

'Wг=${controllers['W']!.text}%; '

'Aг=${controllers['A']!.text}%; '

'Qidaf=${controllers['Qdaf']!.text} МДж/кг';

return SingleChildScrollView(

padding: EdgeInsets.all(16.0),

child: Column(

crossAxisAlignment: CrossAxisAlignment.start,

children: [

...controllers.entries.map((entry) {

final isV = entry.key == 'V';

final isQ = entry.key == 'Qdaf';

return Padding(

padding: const EdgeInsets.symmetric(vertical: 6.0),

child: TextField(

controller: entry.value,

decoration: InputDecoration(

labelText: isV

? '${entry.key} (mg/kg)'

: isQ

? '${entry.key} (МДж/кг)'

: '${entry.key}, %',

border: OutlineInputBorder(),

),

keyboardType: TextInputType.*number*,

),

);

}).toList(),

SizedBox(height: 16.0),

ElevatedButton(

onPressed: calculateTask2,

child: Text('Calculate'),

),

SizedBox(height: 24.0),

if (qWork != null) ...[

Text(

'2. Для складу горючої маси мазуту, що задано:\n$inputComposition',

style: TextStyle(fontWeight: FontWeight.*bold*),

),

SizedBox(height: 8),

Text(

'2.1. Склад робочої маси мазуту становитиме:\n'

' Hр=${hWork?.toStringAsFixed(2)}%; '

'Cр=${cWork?.toStringAsFixed(2)}%; '

'Sр=${sWork?.toStringAsFixed(2)}%; '

'Oр=${oWork?.toStringAsFixed(2)}%; '

'Vр=${vWork?.toStringAsFixed(2)} мг/кг; '

'Wр=${wWork?.toStringAsFixed(2)}%; '

'Aр=${aWork?.toStringAsFixed(2)}%',

),

SizedBox(height: 8),

Text(

'2.2. Нижча теплота згоряння мазуту на робочу масу: '

'${qWork?.toStringAsFixed(2)} МДж/кг',

),

SizedBox(height: 32),

],

],

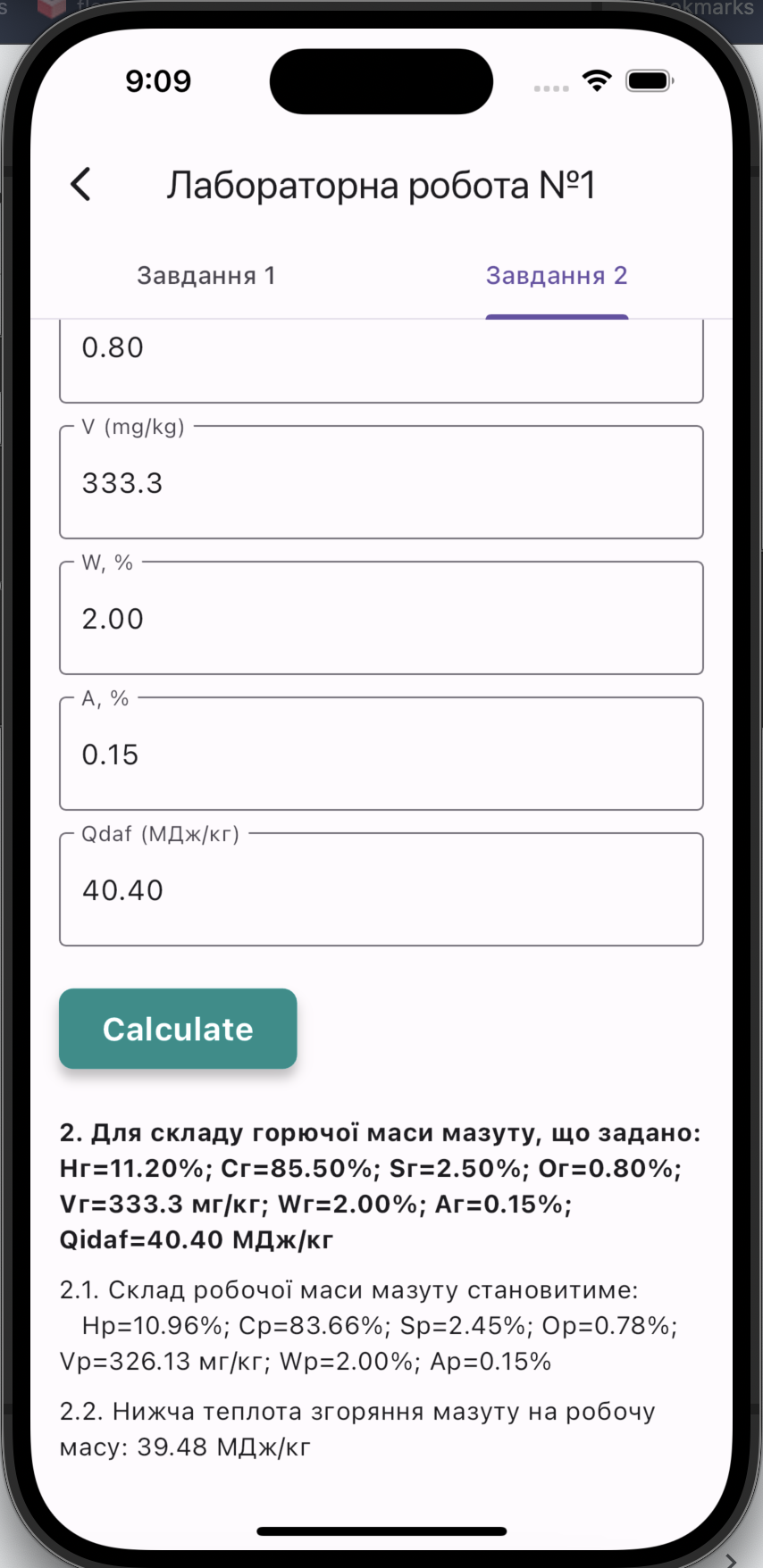
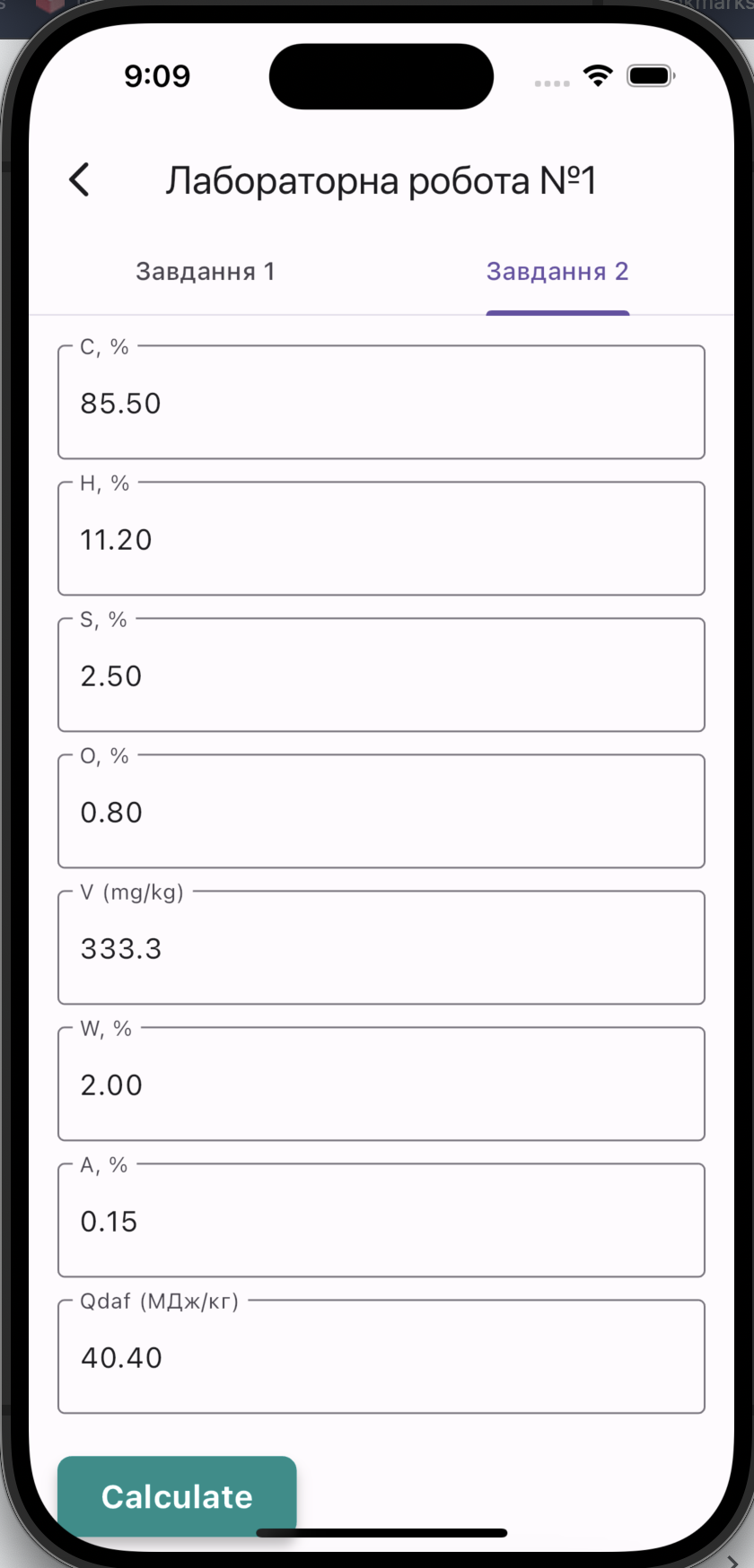
),

);

}

}

Результат перевірки на контрольному прикладі:



**Висновок:**

У даній роботі було реалізовано кросплатформний застосунок, який обчислює параметри складу сухої та горючої маси палива, а також визначає нижчу теплоту згоряння.

**Основні висновки порівняння Flutter та Kotlin:**

* **Універсальність:**Flutter дозволяє використовувати єдиний код для розробки додатків на різних платформах — Android, iOS, Windows, Linux та Web. На відміну від нього, Kotlin (з Jetpack Compose) в основному орієнтований на Android.
* **Розробка інтерфейсу:**Завдяки вбудованим віджетам Flutter спрощує та пришвидшує створення UI, що дозволяє розробляти сучасні інтерфейси швидше.
* **Продуктивність:**Kotlin, маючи нативну підтримку Android, може забезпечувати вищу продуктивність у великих або специфічно орієнтованих на Android проєктах.
* **Гнучкість та прототипування:**Flutter надає більше можливостей для швидкого створення прототипів та адаптації інтерфейсів до різних платформ, що є важливим для кросплатформної розробки.

Використання Flutter в даному проєкті дозволило створити універсальне рішення, здатне працювати як на мобільних, так і на десктопних операційних системах. Це робить його зручним інструментом для розробників, які прагнуть швидко та ефективно впроваджувати кросплатформні рішення, порівняно з традиційним підходом на основі Kotlin.