|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

|  |  |
| --- | --- |
| ФАКУЛЬТЕТ | «Информатика и системы управления» |

|  |  |
| --- | --- |
| КАФЕДРА | «Теоретическая информатика и компьютерные технологии» |

**Отчет по лабораторной работе № 4**

**«Создание меню навигации в AppBar и использование AnimationController»**

***по курсу***

***«Разработка мобильных приложений»***

Выполнил:

Студент группы ИУ9-72Б

Караник А.А.

Проверил:

Посевин Д.П.

1. *г.*

## Цель

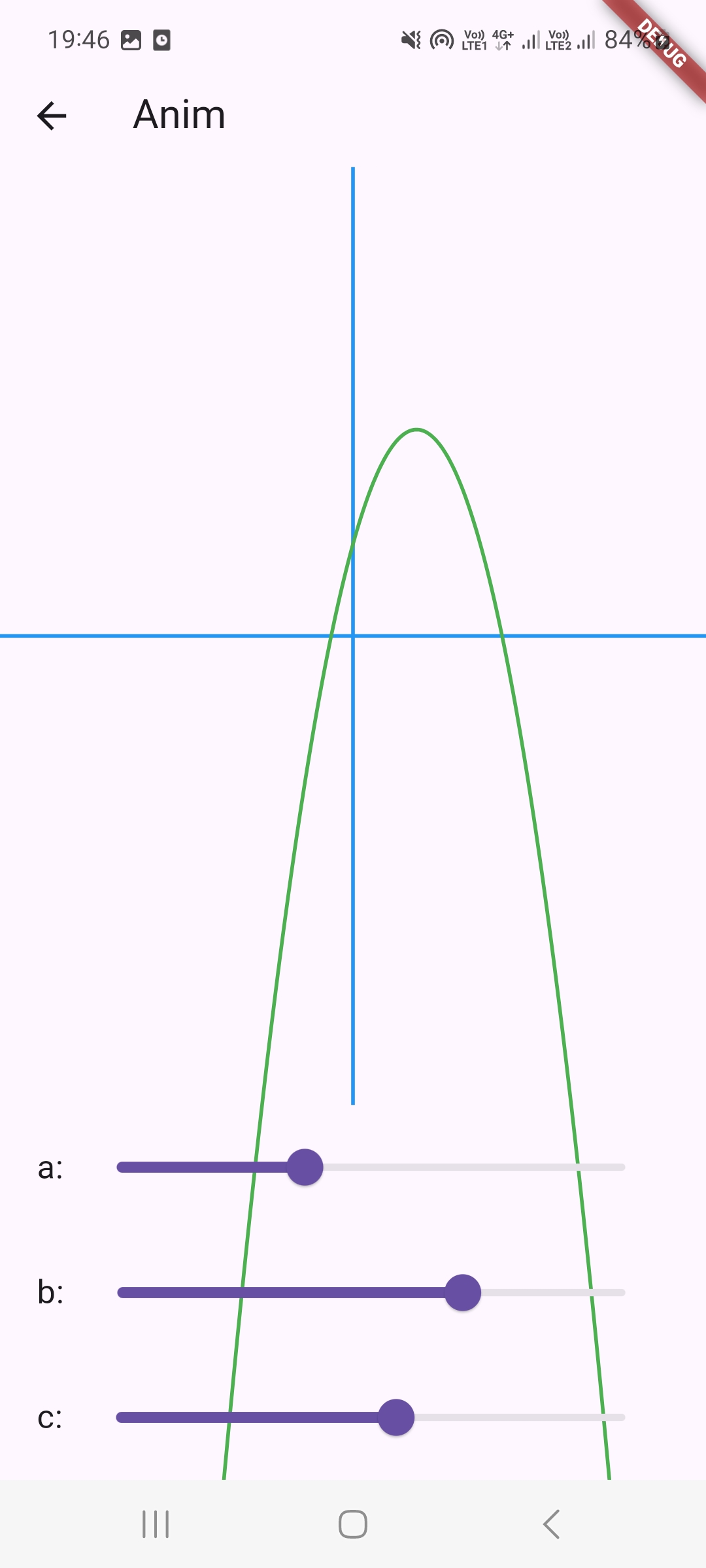
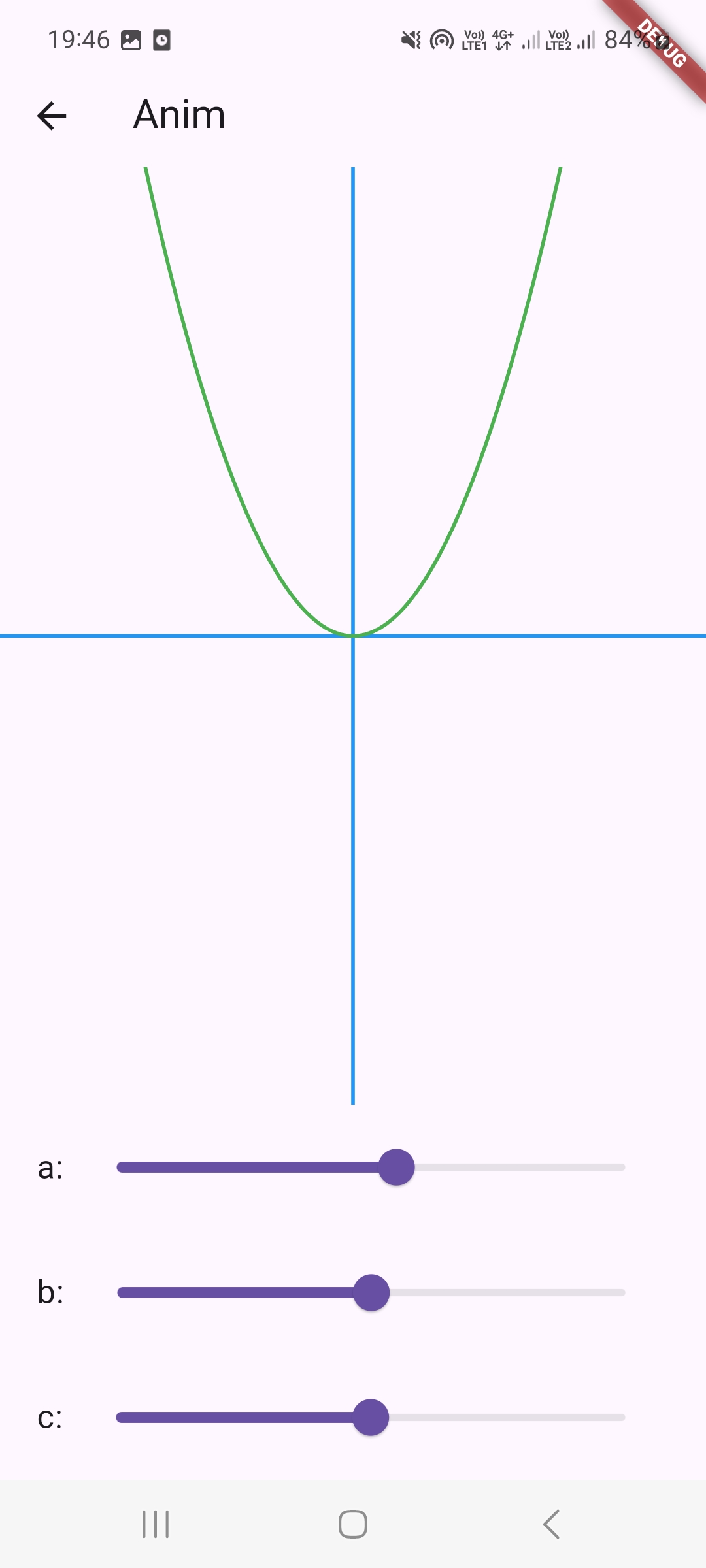
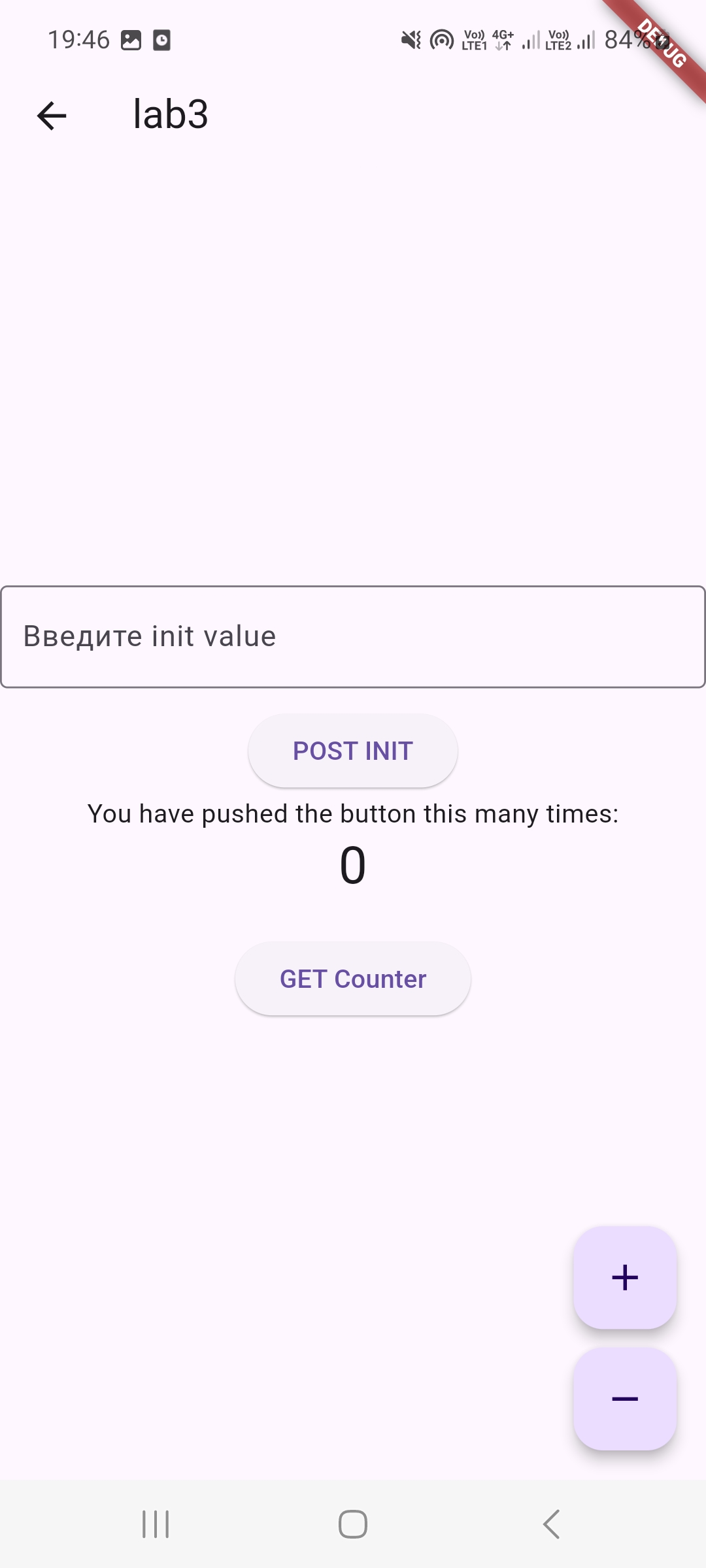
Цель лабораторной работы состоит в том, чтобы научиться создавать выпадающее меню в AppBar для навигации по предыдущим лабораторным работам, а также реализовать экран с использованием AnimationController, доступный через это меню.

## Практическая реализация

Код приложения:

import 'package:flutter/material.dart';  
import 'package:http/http.dart' as http;  
import 'dart:math';  
import 'dart:convert';  
  
void main() {  
 runApp(MyApp());  
}  
  
class MyApp extends StatelessWidget {  
 @override  
 Widget build(BuildContext context) {  
 return MaterialApp(  
 title: 'lab4',  
 home: HomeScreen(),  
 );  
 }  
}  
  
class HomeScreen extends StatefulWidget {  
 @override  
 \_HomeScreenState createState() => \_HomeScreenState();  
}  
  
class \_HomeScreenState extends State<HomeScreen> {  
 @override  
 Widget build(BuildContext context) {  
 return Scaffold(  
 appBar: AppBar(  
 title: Text('lab4'),  
 actions: [  
 PopupMenuButton<String>(  
 onSelected: (value) {  
 if (value == 'lab2') {  
 Navigator.*push*(  
 context,  
 MaterialPageRoute(builder: (context) => Lab2Screen()),  
 );  
 } else if (value == 'lab3') {  
 Navigator.*push*(  
 context,  
 MaterialPageRoute(builder: (context) => Lab3Screen()),  
 );  
 } else if (value == 'anim') {  
 Navigator.*push*(  
 context,  
 MaterialPageRoute(builder: (context) => AnimScreen())  
 );  
 }  
 },  
 itemBuilder: (BuildContext context) {  
 return [  
 PopupMenuItem(  
 value: 'lab2',  
 child: Text('Lab 2'),  
 ),  
 PopupMenuItem(  
 value: 'lab3',  
 child: Text('Lab 3'),  
 ),  
 PopupMenuItem(  
 value: 'anim',  
 child: Text('Anim'),  
 ),  
 ];  
 },  
 ),  
 ],  
 ),  
 body: Center(  
 child: Text('Выберите лабораторную работу из меню'),  
 ),  
 );  
 }  
}  
  
class Lab2Screen extends StatefulWidget {  
 @override  
 \_Lab2ScreenState createState() => \_Lab2ScreenState();  
}  
  
class \_Lab2ScreenState extends State<Lab2Screen> {  
 String response = "Здесь будет ответ";  
 bool isSwitched = false;  
  
 Future<void> requestOff() async {  
 final response = await http.get(Uri.*parse*('http://iocontrol.ru/api/sendData/karanik/value/0'));  
  
 if (response.statusCode == 200) {  
 setState(() {  
 this.response = jsonDecode(response.body).toString();  
 });  
 } else {  
 setState(() {  
 this.response = 'Failed';  
 });  
 }  
 }  
  
 Future<void> requestOn() async {  
 final response = await http.get(Uri.*parse*('http://iocontrol.ru/api/sendData/karanik/value/1'));  
  
 if (response.statusCode == 200) {  
 setState(() {  
 this.response = jsonDecode(response.body).toString();  
 });  
 } else {  
 setState(() {  
 this.response = 'Failed';  
 });  
 }  
 }  
  
 @override  
 Widget build(BuildContext context) {  
 return Scaffold(  
 appBar: AppBar(  
 title: Text("lab2"),  
 ),  
 body: Padding(  
 padding: const EdgeInsets.all(16.0),  
 child: Column(  
 mainAxisAlignment: MainAxisAlignment.center,  
 children: <Widget>[  
 // Переключатель on/off  
 Row(  
 mainAxisAlignment: MainAxisAlignment.center,  
 children: [  
 Text("OFF"),  
 Switch(  
 value: isSwitched,  
 onChanged: (value) {  
 setState(() {  
 isSwitched = value;  
 if (isSwitched) {  
 requestOn();  
 } else {  
 requestOff();  
 }  
 });  
 },  
 ),  
  
 Text("ON"),  
 ],  
 ),  
 SizedBox(height: 16),  
 Text(response),  
 ],  
 ),  
 ),  
 );  
 }  
}  
  
class Lab3Screen extends StatefulWidget {  
 @override  
 \_Lab3ScreenState createState() => \_Lab3ScreenState();  
}  
  
class \_Lab3ScreenState extends State<Lab3Screen> {  
 int \_counter = 0;  
 String \_serverResponse = '';  
 final TextEditingController \_textController = TextEditingController();  
  
 void \_incrementCounter() {  
 setState(() {  
 \_counter++;  
 });  
 \_sendCounterToServer();  
 }  
  
 void \_decrementCounter() {  
 setState(() {  
 \_counter--;  
 });  
 \_sendCounterToServer();  
 }  
  
 // Отправка значения на сервер (POST)  
 Future<void> \_sendCounterToServer() async {  
 final url = Uri.*parse*('http://194.67.88.154:8100/$\_counter');  
 try {  
 final response = await http.post(url);  
  
 if (response.statusCode == 200) {  
 setState(() {  
 \_serverResponse = 'Значение отправлено: $\_counter';  
 });  
 } else {  
 setState(() {  
 \_serverResponse =  
 'Не удалось отправить значение. Сервер ответил кодом статуса: ${response.statusCode}';  
 });  
 }  
 } catch (e) {  
 setState(() {  
 \_serverResponse = 'Ошибка отправки значения: $e';  
 });  
 }  
 }  
  
 Future<void> \_sendValueToServer(int value) async {  
 final url = Uri.*parse*('http://194.67.88.154:8100/$value');  
 try {  
 final response = await http.post(url);  
  
 if (response.statusCode == 200) {  
 setState(() {  
 \_serverResponse = 'Значение отправлено: $value';  
 \_counter = value;  
  
 });  
 } else {  
 setState(() {  
 \_serverResponse =  
 'Не удалось отправить значение. Сервер ответил кодом статуса: ${response.statusCode}';  
 });  
 }  
 } catch (e) {  
 setState(() {  
 \_serverResponse = 'Ошибка отправки значения: $e';  
 });  
 }  
 }  
  
 // Получение значения с сервера (GET)  
 Future<void> \_getValueFromServer() async {  
 final url = Uri.*parse*('http://194.67.88.154:8100');  
 try {  
 final response = await http.get(url);  
  
 if (response.statusCode == 200) {  
 setState(() {  
 \_counter = int.*parse*(response.body);  
 \_serverResponse = 'Счетчик обновлен с сервера: $\_counter';  
 });  
 } else {  
 setState(() {  
 \_serverResponse =  
 'Не удалось получить значение. Сервер ответил кодом статуса: ${response.statusCode}';  
 });  
 }  
 } catch (e) {  
 setState(() {  
 \_serverResponse = 'Ошибка получения значения: $e';  
 });  
 }  
 }  
  
 @override  
 Widget build(BuildContext context) {  
 return Scaffold(  
 appBar: AppBar(  
 title: Text("lab3")  
 ),  
 body: Center(  
 child: Column(  
 mainAxisAlignment: MainAxisAlignment.center,  
 children: <Widget>[  
 const SizedBox(height: 20),  
 TextField(  
 controller: \_textController,  
 decoration: const InputDecoration(  
 labelText: 'Введите init value',  
 border: OutlineInputBorder(),  
 ),  
 ),  
 const SizedBox(height: 10),  
 ElevatedButton(  
 onPressed: () {  
 \_sendValueToServer(int.*parse*(\_textController.text));  
 },  
 child: const Text('POST INIT'),  
 ),  
 const Text(  
 'You have pushed the button this many times:',  
 ),  
 Text(  
 '$\_counter',  
 style: Theme.*of*(context).textTheme.headlineMedium,  
 ),  
 const SizedBox(height: 20),  
 ElevatedButton(  
 onPressed: \_getValueFromServer,  
 child: const Text('GET Counter'),  
 ),  
 const SizedBox(height: 20),  
 Text(  
 \_serverResponse,  
 style: const TextStyle(color: Colors.*lightGreen*),  
 ),  
 ],  
 ),  
 ),  
 floatingActionButton: Column(  
 mainAxisAlignment: MainAxisAlignment.end,  
 children: <Widget>[  
 FloatingActionButton(  
 onPressed: \_incrementCounter,  
 tooltip: 'Increment',  
 child: const Icon(Icons.*add*),  
 ),  
 const SizedBox(height: 10),  
 FloatingActionButton(  
 onPressed: \_decrementCounter,  
 tooltip: 'Decrement',  
 child: const Icon(Icons.*remove*),  
 ),  
 ],  
 ),  
 );  
 }  
}  
  
class AnimScreen extends StatefulWidget {  
 @override  
 \_AnimScreenState createState() => \_AnimScreenState();  
}  
  
class \_AnimScreenState extends State<AnimScreen>  
 with SingleTickerProviderStateMixin {  
 double a = 1;  
 double b = 0;  
 double c = 0;  
  
 late AnimationController \_controller;  
  
 @override  
 void initState() {  
 super.initState();  
 \_controller = AnimationController(  
 vsync: this,  
 duration: Duration(seconds: 1)  
 )..repeat();  
 }  
  
 @override  
 void dispose() {  
 \_controller.dispose();  
 super.dispose();  
 }  
  
 @override  
 Widget build(BuildContext context) {  
 return Scaffold(  
 appBar: AppBar(  
 title: Text('Anim'),  
 ),  
 body: Column(  
 children: [  
 Expanded(  
 child: AnimatedBuilder(  
 animation: \_controller,  
 builder: (context, child) {  
 return CustomPaint(  
 painter: ParabolaPainter(a, b, c),  
 child: Container(),  
 );  
 },  
 ),  
 ),  
 \_buildSlider('a', a, -10, 10, (val) => setState(() => a = val)),  
 \_buildSlider('b', b, -10, 10, (val) => setState(() => b = val)),  
 \_buildSlider('c', c, -10, 10, (val) => setState(() => c = val)),  
 ],  
 ),  
 );  
 }  
  
 Widget \_buildSlider(String label, double value, double min, double max, ValueChanged<double> onChanged) {  
 return Padding(  
 padding: const EdgeInsets.symmetric(horizontal: 20, vertical: 10),  
 child: Row(  
 children: [  
 Text('$label: ', style: TextStyle(fontSize: 18)),  
 Expanded(  
 child: Slider(  
 value: value,  
 min: min,  
 max: max,  
 divisions: 100,  
 label: value.toStringAsFixed(2),  
 onChanged: onChanged,  
 ),  
 ),  
 ],  
 ),  
 );  
 }  
}  
  
class ParabolaPainter extends CustomPainter {  
 final double a;  
 final double b;  
 final double c;  
  
 ParabolaPainter(this.a, this.b, this.c);  
  
 @override  
 void paint(Canvas canvas, Size size) {  
 final paint = Paint()  
 ..color = Colors.*blue* ..strokeWidth = 2  
 ..style = PaintingStyle.stroke;  
  
 final centerX = size.width / 2;  
 final centerY = size.height / 2;  
  
 canvas.drawLine(Offset(0, centerY), Offset(size.width, centerY), paint);  
 canvas.drawLine(Offset(centerX, 0), Offset(centerX, size.height), paint);  
  
 paint.color = Colors.*green*;  
  
 final path = Path();  
 for (double x = -centerX; x <= centerX; x += 1) {  
 double y = a \* pow(x / 50, 2) + b \* (x / 50) + c;  
 if (x == -centerX) {  
 path.moveTo(centerX + x, centerY - y \* 50);  
 } else {  
 path.lineTo(centerX + x, centerY - y \* 50);  
 }  
 }  
  
 canvas.drawPath(path, paint);  
 }  
  
 @override  
 bool shouldRepaint(covariant CustomPainter oldDelegate) {  
 return true;  
 }  
}

## Результаты



## Вывод

В ходе выполнения лабораторной работы был успешно создан функциональный AppBar с выпадающим меню для навигации. Также реализован экран с анимацией, управляемый через AnimationController. Это позволило лучше понять механику навигации и работы с анимациями в Flutter.