Лабораторийн тайлан №10

Flutter фрэймворкын үндсэн ойлголт ба түүний хэрэглээ

МТЭС, МКУТ, Программ хангамж

Г.Батням, 22B1NUM5578

1.Ажлын зорилго

Энэ лабораторийн ажлаар flutter фрэймворкын үндсэн ойлголтуудыг ойлгож эзэмшин өгөгдсөн 4 хэсэг даалгавруудыг зааврын дагуу хийж гүйцэтгэнэ.

2.Онолын судалгаа

**ListView –** Нийт дэлгэцийн хувьд доошоо цувсан байдлаар list буюу жагсаалтыг үүсгэдэг.Элементүүдийн хувьд **leading** буюу толгой хэсэг түүний дараа **title** буюу гарчиг, **subtitle** тухайн list – ийн агуулгыг, **trailing** нь төгсгөл хэсэгт юу байхыг тус тус тодорхойлж өгдөг. **scrollDirection: Axis.vertical** мөн listview – д үүнийг ашиглаад багтахгүй нийт элементийг scroll хийх боломжтой.Зургийн хувьд **reverse** – ийг ашигласнаар өмнөх гарч ирснээс өөр дарааллаар гарч ирдэг.Харин нэг ListTile – ийн хувьд тухайн элементүүд дэлгэцийн доод хэсэгт дарааллын хувьд reverse хийгдсэн байдлаар гарч ирдэг.

**ListView.builder –** Олон ListTile үүсгэх тохиолдолд ашиглагддаг ба дотроо **itemCount** хувьсагчийг агуулдаг ба энэ нь нийт хэдэн хувь үүсгэхийг тодорхойлдог.**itemBuilder** функц нь **(BuildContext context, int index)** параметрүүдийг авдаг ба ListTile үүсгэх процессоо хэрэгжүүлж буцаадаг.Нийт үүсгэх тоог зааж өгөөгүй тохиолдолд Grid – үүдийг үүсгэсээр байх болно.

**GridView –** Нийт байгаа элементүүдийг grid байдлаар дүрсэлдэг.GridView.Count нь тухайн grid нь нэг мөрөндөө нийт хэдэн элементийг багтаахыг тодорхойлдог.GridView.Count нь scroll автоматаар хийгддэг.**GridView** босоо чиглэлд гүйдэг бол, **CrossAxisAlignment.stretch** нь нэг мөрөнд байгаа элемент бүрийг өөрт оногдсон баганын өргөнийг бүхэлд нь эзлэх хүртэл хэвтээ чиглэлд сунгана.Тухайн 1 мөрөнд агуулах элементийн тоогоор дэлгэцийн нийт уртыг хуваадаг.

**GridView.builder –** Олон gridView үүсгэх тохиолдолд хялбарчлах зорилгоор ашигладаг.  
 ***signature:  
 body: GridView.builder(***

***gridDelegate: const SliverGridDelegateWithFixedCrossAxisCount(***

***crossAxisCount: 2,***

***),***

***itemCount: 20,***

***itemBuilder: (BuildContext context, int index) {***

***return GridTile(child: Center(child: Text("Item $index")));***

***},***

***),itemCount – ийн утга нь ашиглагдаж байгаа list – ийн нийт тоотой таарч байх ёстой.***

**Assets –** Өөрийн төсөлд хэрэглэгдэх зураг, фонт гэх мэт зүйлсийг Assets folder – т оруулан хийж түүнийгээ ашигладаг.**Үүнд: Зургууд(jpeg, png, gif), Фонтууд, Дуу(), Видео(жижиг файл), мэдээлэл(json, xml бусад форматлагдсан өгөгдлүүд)**

**Local Image –** Өөрийн төслийн дотоод зургийг локал зураг гэнэ.Үүнийг хэрэглэхийн тулд pupsbec.yaml файлын assets: - аа тохируулж өгдөг. Кодод хэрэглэхдээ Image.assets(“assets/image.png”).

**Network Image –** Image.network(“https://picsum.photos/250?image=9”) ашиглана.

**Cashed Image –** Анх удаа татаж авсны дараа локал санах ойд хадгалагддаг зураг юм. Үүнийг ашигласнаар ачаалах хугацаа багасах, сүлжээний ашиглалт багасах, хэрэгжүүлэлтийг сайжруулах зэрэг давуу талууд үүснэ.**flutter pub add cached\_network\_image**

**Дахин ашиглах -** Хэрэв кэшлэгдсэн зургийг **дахиад шууд таталгүйгээр** ашиглахыг хүсвэл flutter\_cache\_manager-ийг ашиглан доорх байдлаар дуудаж болно.

***Жишээ нь:***

***import'package:flutter\_cache\_manager/flutter\_cache\_manager.dart';***

***void loadCachedImage(String url) async {***

***final file = await DefaultCacheManager().getSingleFile(url);***

***print('Cached file path: ${file.path}');***

***}***

***Тайлбар: placeholder: (context, url) => CircularProgressIndicator(): Энэ нь зураг татагдаж дуусах хүртэл хэрэглэгчид харуулах түр зуурын виджетийг тодорхойлж байна.* *errorWidget: (context, url, error) => Icon(Icons.error): Энэ нь зураг татахад ямар нэгэн алдаа гарсан тохиолдолд хэрэглэгчид харуулах виджетийг тодорхойлж байна.***

**Cache data delete - await DefaultCacheManager().emptyCache();** энэ командыг бичин cached хадгалагдсан бүх зургуудыг устгана.

**Local Fonts –** Интернэтгүй орчинд ашиглах боломжтой font – ийг pubspec.yaml файлдаа **fonts: -family: Roboto fonts: -assets:** гэх мэт байдлаар тодорхойлох ёстой.Ашиглахдаа тухайн family – ийн нэрийг ашиглан хэрэгжүүлдэг. **fontFamily: 'Roboto’** Нэрийг зөв тодорхойлж өгөх шаардлагатай**.**

**Google fonts –** Интернэтээс буюу google – с font татан ашиглах боломжтой. **flutter pub add google\_fonts** энэ командыг заавал оруулдаг ингэснээр тухайн төсөлд google font – уудыг ашиглах боломжтой болно.Энэ команд нь pubsec.yaml файлд автоматаар dependency – г буюу санг нэмж өгдөг.Ашиглахдаа **style: GoogleFonts.roboto()** гэх жишээгээр ашиглана.Мөн түүнийгээ **import** хийж өгөх хэрэгтэй.

**Local Json data –** Өөрийн assets дотор байгаа json датаг ашиглахдаа local service үүсгэн түүгээрээ өөрийн local дотор хадгалагдаж байгаа файлынхаа замаар дамжин өгөгдөлдөө ханддаг.

**Future –** Энэ нь ирээдүйд хийгдэх үйлдлийн асинхрон үйлдлийг илэрхийлдэг.

**rootBundle –** Дотоод файлаас өгөгдөл унших тохиолдолд ашиглагддаг.   
***// json.decode() is used to convert JSON String to JSON Map  
final jsonResponse = json.decode(jsonString);***

**Builder snapshot –** builder функцын snapshot параметр future ажиллагааны төлвийг snapshot объект дамжуулан builder функцэд мэдэгддэг.

**Жишээ нь: snapshot – д Өгөгдөл байгаа эсэхийг шалгадаг.**

***if (snapshot.hasData) {}***

**Online json data - flutter pub add http** командыг ажиллуулан интернэтээс өгөгдөл датан авах боломжтой болно.Хэрэгжүүлэхдээ мөн future async хэрэгжүүлэн түүн дотроо http хүсэлтээ илгээн хариу хүлээн авч түүнийхээ ямар утга буцан ирснээс хамаарч дата авах эсэх нь тодорхойлогддог.Хариуг шалгахдаа statusCode – ийг жишдэг.

**import 'package:http/http.dart' as http;** - Ингэснээр http хүсэлтийг хэрэгжүүлэх протоколоо оноож байна.

**Local audio - flutter pub add audioplayers** энэ командын тусламжтайгаар local орчинд байгаа audio – г ашиглах боломжтой болно.AudioPlayer – ийн байгуулагчаар объектоо үүсгээд үүсгэсэн объектынхоо play үйлдлийг ашиглан AssetSource болон UrlSource ашиглан local дуу болон интернэт дэх дууг тоглуулж зогсоож ажиллуулах боломжтой.

**Drawer** - (шуурхай самбар) нь дэлгэцийн хажуугаас гарч ирдэг, ихэвчлэн цэсний сонголтууд эсвэл навигацийн холбооснуудаар дүүрэн байдаг бөгөөд аппликешны өөр өөр хэсгүүдийн хооронд шилжих боломжийг олгодог.Scaffold drawer – ийг ашиглан хэрэгжүүлдэг.

**Snackbar -** (мэдэгдлийн самбар) нь дэлгэцийн доод хэсэгт гарч ирдэг мессежний самбар юм. Үүнийг **хадгалагдсан** эсвэл **устгагдсан** гэх мэт богино мэдэгдлүүдийг харуулах, мөн хэрэглэгчдэд **буцаах** эсвэл **дахин оролдох** зэрэг үйлдлүүдийг хийх боломж олгоход ашигладаг.

**Жишээ нь:**

**ScaffoldMessenger.of(context).showSnackBar(**

**SnackBar(**

**content: Text('Message deleted'),**

**backgroundColor: Colors.blue,**

**duration: Duration(seconds: 10),*//нийт үргэлжлэх хугацаа***

**action: SnackBarAction(**

**label: 'UNDO',**

**onPressed: () {**

**// Perform some action**

**},**

**),//*button хэрэгжүүлэх үед***

**),**

**);**

***Жишээ нь:***

***ScaffoldMessenger.of(context).showSnackBar(***

***const SnackBar(***

***content: Text("Account created successfully"),***

***duration: Duration(seconds:5),***

***backgroundColor:Colors.green,***

***),***

**Bottom Navigation Bar** - (доод навигацийн самбар) нь дэлгэцийн доод хэсэгт байрлах материал виджет бөгөөд аппликешны өөр өөр хэсгүүд эсвэл функцүүдийн хооронд хялбар навигаци хийх зориулалттай олон таб-уудыг харуулдаг.

**Material Button (Материал товчлуур)** - Ихэвчлэн материал дизайныг ашигладаг товчлуур.

**Гол Ойлголтуудын Нэгтгэл:**

* **onPressed:** Товчлуур дарахад ажиллах үйлдлийг тодорхойлдог чухал параметр. Хэрэв null бол товчлуур идэвхгүй болно.
* **child:** Товчлуурын дотор байрлах виджет (текст, зураг, эсвэл бусад виджетүүд).
* **color:** Товчлуурын дэвсгэр өнгө.
* **textColor:** Товчлуурын текстийн өнгө.
* **shape:** Товчлуурын хэлбэрийг тохируулах (жишээ нь, дугуйрсан ирмэг).  
  ***shape: RoundedRectangleBorder(borderRadius: BorderRadius.circular(18.0),),***
* **elevation:** Товчлуурын сүүдрийн өндөр.
* **Ink болон BoxDecoration:** Градиент болон бусад чимэглэлийг товчлуурын дэвсгэр дээр зурах арга.Өнгө холин хэрэгжүүлж болдог.  
  ***gradient: const LinearGradient(colors: [Colors.blue, Colors.red]),***
* **Row болон Column:** Товчлуурын дотор олон виджетийг байрлуулах зориулалттай макет виджетүүд.
* **SizedBox:** Виджетүүдийн хооронд зай үүсгэхэд ашиглагддаг.

**Elevated Button (Өргөгдсөн товчлуур)** - Чухал үйлдлүүдэд зориулсан сүүдэртэй товчлуур.

**Гол Ойлголтуудын Нэгтгэл:**

* **onPressed**: Товчлуур дарахад ажиллах үйлдлийг тодорхойлдог чухал параметр. Хэрэв null бол товчлуур идэвхгүй болно.
* **child**: Товчлуурын дотор байрлах виджет (текст, зураг, эсвэл бусад виджетүүд).
* **style: backgroundColor**: Товчлуурын дэвсгэр өнгө.***backgroundColor: MaterialStateProperty.all(Colors.green),***
* **style: textStyle**: Товчлуурын текстийн өнгө болон стиль (фонтын хэмжээ гэх мэт).   
  ***textStyle: MaterialStateProperty.all(TextStyle(fontSize: 20),),***
* **style: shape**: Товчлуурын хэлбэрийг тохируулах (жишээ нь, дугуйрсан ирмэг). ***shape:MaterialStateProperty.all<RoundedRectangleBorder>(RoundedRectangleBorder(orderRadius: BorderRadius.circular(18.0),),),***
* **style: elevation**: Товчлуурын сүүдрийн өндөр.
* **ElevatedButton.icon:** Зураг болон текст агуулсан товчлуур үүсгэхэд ашиглагддаг тусгай конструктор.   
  ***style: ButtonStyle(elevation: MaterialStateProperty.all(10),),***
* **MaterialStateProperty:** Товчлуурын янз бүрийн төлөвт (энгийн, дарагдсан, идэвхгүй) зориулсан утгыг тодорхойлоход ашиглагддаг.

**Text Button (Текст товчлуур)** - Ач холбогдол багатай үйлдлүүдэд зориулсан хавтгай товчлуур.

**Гол Ойлголтуудын Нэгтгэл:**

* **onPressed**: Товчлуур дарахад ажиллах үйлдлийг тодорхойлдог чухал параметр. Хэрэв null бол товчлуур идэвхгүй болно.
* **TextButton.icon:** Тухайн текстийг icon – ий хамт нэг бүхэл болгох бол үүнийг ашигладаг. ***TextButton.icon(onPressed: () {},icon: ,label: )***
* **style: TextButton.styleFrom:** Өнгө болоод загварыг тодорхойлох бол ашиглана. ***style: TextButton.styleFrom(foregroundColor: Colors.black, backgroundColor: Colors.yellow,),***
* **underline:** Текст доогуур зураастай байдлаар button илэрхийлэх гэж байгаа бол хэрэглэнэ***.textStyle: TextStyle(decoration: TextDecoration.underline),***

**Outlined Button (Хүрээтэй товчлуур)** - Хоёрдогч үйлдлүүдэд зориулсан хүрээтэй товчлуур.

**Гол Ойлголтуудын Нэгтгэл:**

* **onPressed**: Товчлуур дарахад ажиллах үйлдлийг тодорхойлдог чухал параметр. Хэрэв null бол товчлуур идэвхгүй болно.
* **style:** Хүрээ болон тухайн текстийн утгыг тодорхойлох боломжтой. ***style: OutlinedButton.styleFrom( side: BorderSide(color: Colors.green, width: 2), textStyle: TextStyle(color: Colors.green, fontSize: 20), ),***
* **shape:** Хэлбэрийг нь мөн өөрчлөж болдог. ***shape: RoundedRectangleBorder( borderRadius: BorderRadius.circular(18.0), ),***

**Icon Button (Зургийн товчлуур)** - Зурагтай товчлуур.

**Гол Ойлголтуудын Нэгтгэл:**

* **onPressed**: Товчлуур дарахад ажиллах үйлдлийг тодорхойлдог чухал параметр. Хэрэв null бол товчлуур идэвхгүй болно.
* **tooltip:** Тухайн button дээр mouse хүрэнгүүт мэдээлэл нь гарч ирдэг ***tooltip: 'More Info',***

**Floating Action Button (Хөвөгч үйлдлийн товчлуур)** - Дугуй хэлбэртэй гол үйлдлийн товчлуур.

**Гол Ойлголтуудын Нэгтгэл:**

* **onPressed**: Товчлуур дарахад ажиллах үйлдлийг тодорхойлдог чухал параметр. Хэрэв null бол товчлуур идэвхгүй болно.
* **mini: true:** FAB-ын жижиг хувилбарыг үүсгэж болно
* **extended:** Button дотроо олон widget – уудыг байрлуулж өргөтгөж болно. ***floatingActionButton: FloatingActionButton.extended()***

**Dropdown Button (Унах цэстэй товчлуур)** - Жагсаалтаас сонгох товчлуур.   
***Жишээ нь:  
DropdownButton<String>(***

***value: dropdownValue,***

***onChanged: (String? newValue) {***

***setState(() {***

***dropdownValue = newValue!;***

***});***

***},***

***items: <String>['One', 'Two', 'Three', 'Four']***

***.map<DropdownMenuItem<String>>((String value) {***

***return DropdownMenuItem<String>(***

***value: value,***

***child: Text(value),***

***);***

***}).toList(),***

***)***

**Гол Ойлголтуудын Нэгтгэл:**

* **DropdownButton<String>:** DropdownButton виджетийг үүсгэж байна. <String> нь сонгогдох утгын төрлийг зааж байна (энэ тохиолдолд String).
* **value: dropdownValue:** Одоогийн сонгогдсон утгыг хадгалдаг хувьсагч. Энэ утга нь items жагсаалтад байгаа нэгэн утгатай тохирч байх ёстой.
* **onChanged: (String? newValue) { ... }:** Хэрэглэгч жагсаалтаас шинэ утга сонгоход ажиллах функц. newValue нь сонгогдсон шинэ утгыг агуулна. setState() функц нь UI-г шинэчлэхэд ашиглагддаг.
* **items: <String>['One', 'Two', 'Three', 'Four'].map<DropdownMenuItem<String>>((String value) { ... }).toList():** Dropdown-д харагдах сонголтуудын жагсаалтыг үүсгэдэг.
* <String>['One', 'Two', 'Three', 'Four']: Сонголтуудын текстийг агуулсан String төрлийн жагсаалт.
* .map<DropdownMenuItem<String>>((String value) { ... }): Жагсаалтын элемент бүрийг DropdownMenuItem<String> виджетэд хөрвүүлдэг.
* return DropdownMenuItem<String>(value: value, child: Text(value));: DropdownMenuItem нь жагсаалтад харагдах нэг сонголтыг төлөөлдөг. value нь сонгогдсон үед буцаагдах утга, child нь харагдах виджет (энэ тохиолдолд сонголтын тексттэй Text виджет) юм.
* .toList(): map() функцийн үр дүнг List төрөл рүү хөрвүүлдэг.

**Popup Menu Button (Нэмэлт цэсний товчлуур)** - Нэмэлт сонголтуудыг харуулдаг товчлуур.  
***Жишээ нь:  
PopupMenuButton<int>(***

***onSelected: (int result) {***

***setState(() {***

***\_selectedMenu = result;***

***});***

***},***

***itemBuilder: (BuildContext context) => <PopupMenuEntry<int>>[***

***const PopupMenuItem<int>(***

***value: 1,***

***child: Text('About Us'),***

***),***

***const PopupMenuItem<int>(***

***value: 2,***

***child: Text('Contact Us'),***

***),***

***const PopupMenuItem<int>(***

***value: 3,***

***child: Text('Privacy Policy'),***

***),***

***],***

***)***

**Гол Ойлголтуудын Нэгтгэл:**

* **PopupMenuButton<int>:** PopupMenuButton виджетийг үүсгэж байна. <int> нь сонгогдсон цэсний зүйлийн утгын төрлийг зааж байна (энэ тохиолдолд int).
* **onSelected: (int result) { ... }:** Хэрэглэгч цэснээс нэг зүйл сонгоход ажиллах функц. result нь сонгогдсон PopupMenuItem-ийн value-г агуулна. setState() функц нь UI-г шинэчлэхэд ашиглагддаг.
* **itemBuilder: (BuildContext context) => <PopupMenuEntry<int>>[...]:** Цэсний зүйлсийг үүсгэдэг функц. Энэ нь PopupMenuEntry<int> төрлийн виджетүүдийн жагсаалтыг буцаах ёстой.
* <PopupMenuEntry<int>>[...]: Цэсний зүйлсийн жагсаалт.
* const PopupMenuItem<int>(value: 1, child: Text('About Us')): Цэсний нэг зүйлийг төлөөлдөг PopupMenuItem виджет.
  + **value: 1:** Энэ цэсний зүйл сонгогдсон үед onSelected функцэд дамжуулагдах утга.
  + **child: Text('About Us'):** Цэсэнд харагдах виджет (энэ тохиолдолд 'About Us' тексттэй Text виджет). const нь энэ виджет өөрчлөгдөшгүй болохыг зааж байна.

**Form -** Виджет нь текст оруулах талбар, чекбокс, товчлуур зэрэг олон form-ын талбаруудыг нэгтгэж, мэдээлэл оруулах хялбар туршлагыг бий болгоход ашиглагддаг.  
***Жишээ нь:  
final \_formKey = GlobalKey<FormState>();***

***Form(***

***key: \_formKey,***

***child: Column(***

***children: [***

***// Form-ын талбарууд энд байна***

***TextFormField(***

***decoration: const InputDecoration(***

***labelText: 'First Name',***

***hintText: 'Enter your first name',***

***),***

***validator: (value) {***

***if (value == "") {***

***return 'Please enter your first name';***

***}***

***return null;***

***},***

***),***

***TextFormField(***

***decoration: const InputDecoration(***

***labelText: 'Last Name',***

***hintText: 'Enter your last name',***

***),***

***validator: (value) {***

***if (value == "") {***

***return 'Please enter your last name';***

***}***

***return null;***

***},***

***),***

***const SizedBox(height: 10),***

***ElevatedButton(***

***onPressed: () {***

***if (\_formKey.currentState!.validate()) {***

***// Илгээх логикийг энд гүйцэтгөнө***

***}***

***},***

***child: const Text('Submit'),***

***),***

***],***

***),***

***)***

**Гол Ойлголтуудын Нэгтгэл:**

* **final \_formKey = GlobalKey<FormState>();:** Form-ын төлөвийг удирдах зориулалттай GlobalKey үүсгэж байна.
* **key: \_formKey:** Form виджетэд үүсгэсэн GlobalKey-г холбож байна.
* **validator: (value) { ... }:** Текст оруулах талбарын утгыг баталгаажуулах функц. Хэрэв утга нь хүчингүй бол алдааны мессежийг буцаана, эсрэг тохиолдолд null буцаана.
* **ElevatedButton(onPressed: () { ... }):** Form-ыг илгээх товчлуур.
* **if (\_formKey.currentState!.validate()) { ... }:** Form-ын төлөвийн validate() методыг дуудаж бүх талбарын баталгаажуулалтыг шалгана. Хэрэв бүх талбар хүчинтэй бол true буцаана.

**Email validation :  
 *validator: (value) {***

***if (value == null || value.isEmpty) {***

***return 'Please enter your email';***

***} else if (!value.contains('@')) {***

***return 'Please enter a valid email';***

***}***

***return null;***

***},***

**Password validation:  
*validator: (value) {***

***if (value == null || value.isEmpty) {***

***return 'Please enter your password';***

***} else if (value.length < 6) {***

***return 'Password must be at least 6 characters';***

***}***

***return null;***

***},***

**TextFormField -** Энэ нь текстэн мэдээллийг цуглуулах, баталгаажуулах, хадгалахад хялбар болгодог.

**Гол Ойлголтуудын Нэгтгэл:**

* **inputFormatters:** Авах утгын төрлийг тодорхойлж өгдөг. ***inputFormatters: [  
  FilteringTextInputFormatter.digitsOnly,  
  LengthLimitingTextInputFormatter(10),  
  ],***
* **Баталгаажуулалттай TextFormField:** validator функц нь хэрэглэгчийн оруулсан утгыг баталгаажуулахад ашиглагддаг.  
  ***validator: (value) { if (value == null || value.isEmpty) { return 'Please enter your email'; } return null; },***
* **Нууц Үгний Талбар:** obscureText: true пропертийг ашиглан оруулж буй текстийг нуух боломжтой (жишээ нь, нууц үг оруулах талбар). ***obscureText: true,***
* **Стильчилсэн TextFormField:** decoration параметрийн дотор border зэрэг пропертиудыг ашиглан TextFormField-ын гаднах үзэмжийг өөрчлөх боломжтой.  
  ***decoration: InputDecoration( border: OutlineInputBorder( borderRadius: BorderRadius.circular(10), ),***
* **Prefix болон Suffix Дүрсүүдтэй TextFormField:** Form – ийн урд болон хойно нь icon байрлуулж болно.   
  ***decoration: InputDecoration(   
  labelText: 'Phone Number',   
  prefixIcon: Icon(Icons.phone),   
  suffixIcon: Icon(Icons.check\_circle),   
  ),***
* **Хамгийн Их Урттай TextFormField:** maxLength пропертийг ашиглан хэрэглэгчийн оруулж болох хамгийн их тэмдэгтийн тоог хязгаарлах боломжтой. ***maxLength: 10,***
* **Тоон гарыг Нээдэг TextFormField:** keyboardType пропертийг TextInputType.number болгож тохируулан фокус авах үед зөвхөн тоон гарыг нээх боломжтой. ***keyboardType: TextInputType.number,***

**Checkbox –** Flutter -т хэрэглэгчдэд хоёр төлөвийн нэгийг сонгох боломжийг олгодог виджет юм: checked (сонгогдсон) эсвэл unchecked (сонгогдоогүй). Үүнийг ихэвчлэн form болон тохиргоонд сонголтыг идэвхжүүлэх эсвэл идэвхгүй болгоход ашигладаг.

***Жишээ нь:***

***List<String> interests = [***

***'Reading',***

***'Music',***

***'Travel',***

***'Sports',***

***'Cooking',***

***];***

***List<String> selectedInterests = [];***

***Column(***

***children: interests.map((String interest) {***

***return CheckboxListTile(***

***title: Text(interest),***

***value: selectedInterests.contains(interest),***

***onChanged: (bool? value) {***

***setState(() {***

***if (value == true) {***

***selectedInterests.add(interest);***

***} else {***

***selectedInterests.remove(interest);***

***}***

***});***

***},***

***);***

***}).toList(),***

***)***

**Гол ойлголтуудын нэгтгэл**

* **value:** Чекбоксын одоогийн төлөвийг (true - сонгогдсон, false - сонгогдоогүй) хадгалдаг логик хувьсагч.
* **onChanged:** Чекбоксын төлөв өөрчлөгдөхөд ажиллах функц. Шинэ төлөвийн утгыг (true эсвэл false) параметрээр дамжуулдаг. setState() функц нь UI-г шинэчлэхэд ашиглагддаг. 
* **checkColor:** Чекбокс сонгогдсон үед харагдах тэмдгийн өнгө.
* **activeColor:** Чекбокс идэвхтэй (сонгогдсон) үеийн дэвсгэр өнгө.
* **CheckboxListTile:** Нэг мөрөнд чекбокс болон тайлбарыг хамтад нь харуулах виджет.
* **title:** Чекбоксын баруун талд харагдах тайлбар виджет (ихэвчлэн Text).
* **value:** Чекбоксын төлөвийг (true эсвэл false) тодорхойлдог логик утга.
* **onChanged:** Чекбоксын төлөв өөрчлөгдөхөд ажиллах функц.

**RadioButton –** Flutter -т бүлгээс зөвхөн нэг сонголтыг сонгох боломжийг олгодог оролтын виджет юм. Checkbox нь олон сонголт хийх боломжтой бол Radio Button нь нэг удаад зөвхөн нэгийг сонгох боломжтой.

***Жишээ нь:  
RadioListTile<Gender>( title: Text(option.text), value: option, groupValue: \_selectedOption, onChanged: (value) { setState(() { \_selectedOption = value; }); }, )***

**Гол Ойлголтуудын Нэгтгэл:**

* **enum ашиглан сонголтуудыг тодорхойлох:** Радио товчлууруудын сонголтуудыг тодорхойлохын тулд enum (enumerated type) ашиглах нь зүйтэй. enum-д сонголтын утга болон харагдах текстийг тодорхойлж болно. ***enum Gender { male("Male"), female("Female"), others("Others"); final String text; const Gender(this.text); }***
* **Сонгогдсон сонголтыг хадгалах төлөвийн хувьсагч:** Сонгогдсон радио товчлуурын утгыг хадгалахын тулд төлөвийн хувьсагч (state variable) үүсгэх шаардлагатай. Энэ хувьсагч нь enum-ийн нэг утгыг агуулна. ***Gender? \_selectedOption = Gender.male;***
* **<Gender>:** RadioListTile-ийн утгын төрлийг зааж байна (энэ тохиолдолд Gender enum).
* **title:** Радио товчлуурын баруун талд харагдах тайлбар виджет (ихэвчлэн Text).
* **value:** Энэ радио товчлуурын утга (энэ тохиолдолд Gender enum-ийн нэг утга).
* **groupValue:** Радио товчлууруудын бүлгийн одоогийн сонгогдсон утга (төлөвийн хувьсагч \_selectedOption). Нэг бүлгийн радио товчлуурууд ижил groupValue-тай байна.
* **onChanged:** Радио товчлуурын сонголт өөрчлөгдөхөд ажиллах функц. Сонгогдсон шинэ утгыг (value) параметрээр дамжуулдаг. setState() функц нь UI-г шинэчлэхэд ашиглагддаг.
* **Радио товчлууруудын бүлгийг үүсгэх:** Нэг бүлгийн радио товчлууруудыг нэгтгэхийн тулд тэдгээрийг ижил groupValue-тай холбож, Column эсвэл Row зэрэг виджетэд байрлуулна.
* **map() ашиглан радио товчлууруудыг динамикаар үүсгэх:** enum-ийн утгуудыг ашиглан map() функцээр дамжуулан радио товчлууруудын жагсаалтыг динамикаар үүсгэх нь код бичихэд хялбар бөгөөд уян хатан байдлыг хангадаг.

**Switch виджет** – нь Flutter-т хоёр төлөвийн хооронд (жишээ нь, асаах/унтраах эсвэл үнэн/худал) шилжихэд ашиглагддаг үндсэн UI бүрэлдэхүүн хэсэг юм.

***Жишээ нь:  
Switch(***

***value: isSwitched,***

***activeTrackColor: Colors.lightGreenAccent, // Идэвхтэй үеийн замын өнгө***

***activeColor: Colors.green, // Идэвхтэй үеийн бөөрөнхий хэсгийн өнгө***

***onChanged: (value) {***

***setState(() {***

***isSwitched = value;***

***});***

***},***

***)  
SwitchListTile(***

***title: Text(key),***

***value: settings[key]!,***

***onChanged: (bool value) {***

***setState(() {***

***settings[key] = value;***

***});***

***},***

***)*Гол Ойлголтуудын Нэгтгэл:**

* **value:** Switch-ийн одоогийн төлөвийг (true - асаалттай, false - унтраалттай) хадгалдаг логик хувьсагч.
* **onChanged:** Switch-ийн төлөв өөрчлөгдөхөд ажиллах функц. Шинэ төлөвийн утгыг (true эсвэл false) параметрээр дамжуулдаг. setState() функц нь UI-г шинэчлэхэд ашиглагддаг.
* **activeTrackColor:** Switch асаалттай үед харагдах замын өнгө.
* **activeColor:** Switch асаалттай үед харагдах бөөрөнхий хэсгийн өнгө. Мөн inactiveTrackColor болон inactiveThumbColor гэсэн идэвхгүй үеийн өнгөнүүд байдаг.
* **SwitchListTile:** Нэг мөрөнд switch болон тайлбарыг хамтад нь харуулах виджет.
* **title:** Switch-ийн зүүн талд харагдах тайлбар виджет (ихэвчлэн Text).
* **value:** Switch-ийн төлөвийг (true эсвэл false) тодорхойлдог логик утга.
* **onChanged:** Switch-ийн төлөв өөрчлөгдөхөд ажиллах функц.

**DateTime Picker** – нь Flutter-т хэрэглэгчдэд диалогоос огноо, цаг эсвэл хоёуланг нь сонгох боломжийг олгодог виджет юм. Энэ нь хурал товлох эсвэл сануулах зэрэг огноо, цаг оруулах шаардлагатай аппликейшнд маш хэрэгтэй.  
***Жишээ нь:  
DateTime? selectedDateTime;***

***Future<void> \_selectDateTime(BuildContext context) async {***

***final DateTime? pickedDate = await showDatePicker(***

***context: context,***

***initialDate: DateTime.now(),***

***firstDate: DateTime(1900),***

***lastDate: DateTime.now(),***

***);***

***if (pickedDate != null) {***

***final TimeOfDay? pickedTime = await showTimePicker(***

***context: context,***

***initialTime: TimeOfDay.now(),***

***);***

***if (pickedTime != null) {***

***setState(() {***

***selectedDateTime = DateTime(***

***pickedDate.year,***

***pickedDate.month,***

***pickedDate.day,***

***pickedTime.hour,***

***pickedTime.minute,***

***);***

***});***

***}***

***}***

***}***

**Гол Ойлголтууд:**

* **Огноо Сонгогч (Date Picker):** showDatePicker функц нь хэрэглэгчдэд огноо сонгох боломжийг олгодог диалог харуулдаг
* **showDatePicker(context: ..., initialDate: ..., firstDate: ..., lastDate: ...):** Огноо сонгох диалогийг харуулдаг асинхрон функц.
* **context:** Тухайн виджетийн контекст.
* **initialDate:** Диалог анх нээгдэхэд сонгогдсон байх огноо.
* **firstDate:** Хэрэглэгчийн сонгох боломжтой хамгийн эртний огноо.
* **lastDate:** Хэрэглэгчийн сонгох боломжтой хамгийн сүүлийн огноо.
* **selectedDate:** Хэрэглэгчийн сонгосон огноог хадгалах DateTime төрлийн төлөвийн хувьсагч.
* **setState(() { selectedDate = picked; });:** Огноо сонгогдсоны дараа UI-г шинэчлэхэд ашиглагддаг.  **showTimePicker(context: ..., initialTime: ...):** Цаг сонгох диалогийг харуулдаг асинхрон функц.
* **context:** Тухайн виджетийн контекст.
* **initialTime:** Диалог анх нээгдэхэд сонгогдсон байх цаг. TimeOfDay.now() нь одоогийн цагийг буцаадаг.
* **selectedTime:** Хэрэглэгчийн сонгосон цагийг хадгалах TimeOfDay төрлийн төлөвийн хувьсагч.
* **setState(() { selectedTime = picked; });:** Цаг сонгогдсоны дараа UI-г шинэчлэхэд ашиглагддаг.

**Navigation -** гэдэг нь хэрэглэгч нэг дэлгэцээс (жишээ нь Home дэлгэц) нөгөө дэлгэц рүү (жишээ нь Contact дэлгэц) шилжих үйл явц юм. Flutter апп-д дэлгэцүүдийг **screens**, **routes**, эсвэл **pages** гэж нэрлэдэг. Flutter-д navigation хийх **3 үндсэн арга** байдаг

1. **Navigator ашиглах**  
   ***Жишээ нь:  
   Navigator.of(context).push(***

***MaterialPageRoute(builder: (context) => SecondScreen()),***

***);***

Энэ код нь **SecondScreen** дэлгэцийг нээж, одоогийн дэлгэц дээр давхарлаж харуулна.**push()** нь шинэ дэлгэцийг stack дээр нэмэх гэсэн үг.  
**Navigator.pop(context); -** Энэ нь одоо харуулж буй дэлгэцийг хааж, өмнөх дэлгэц рүү буцах үйлдэл. Энэ нь **back товч** дарсантай ижил үйлдэл юм.

1. **Named Routes ашиглах  
   Тайлбар:** Route бүрт нэр өгч, түүний дагуу шилжих аргаMaterialApp дээр route-уудыг тодорхойлно. Дараа нь Navigator.pushNamed ашиглан шилждэг: ***routes: { '/profile': (context) => ProfileScreen() }***  
   ***Жишээ нь:  
   MaterialApp(***

***routes: {***

***'/': (context) => FirstScreen(),***

***'/second': (context) => SecondScreen(),***

***},***

***);***

**Navigator.pushNamed(context, '/second');** Энэ код нь second нэртэй дэлгэц рүү шилжихийг заана. ***initialRoute: '/'*** *гэдэг нь апп анх ажиллахад FirstScreen-ийг ачаална гэсэн үг.* ***'/second': (context) => SecondScreen(),*** *SecondScreen – д өгсөн нэр*

1. **Router (жишээ нь: go\_router) ашиглах  
   Тайлбар:** Том, олон үе шаттай апп-д зориулсан илүү уян хатан navigation систем. go\_router, auto\_route гэх мэт пакетууд ашигладаг. Урьдчилсан төлөвлөлт, URL path дээр суурилсан navigation хийж болдог.

**Бусад дэлгэц рүү дата дамжуулах**

1. **Нэрээр нь дамжуулах:  
   Илгээгч:***Navigator.of(context).push(MaterialPageRoute(  
   builder: (context) => GreetScreen(name: \_controller.text),***Хүлээн авагч:** *final String name;  
   const GreetScreen({Key? key, required this.name}) : super(key: key);* байдлаар байгуулагчыг зарлаж болно.
2. **Объект дамжуулах:  
   Илгээгч:** *final product = Product(name: 'Laptop', price: 1000);  
   Navigator.of(context).push(MaterialPageRoute(*

*builder: (context) => ProductScreen(product: product),*

*));***Хүлээн авагч:** *final Product product;  
const ProductScreen({Key? key, required this.product}) : super(key: key);*

1. **Named Route ашиглан текст дамжуулах  
   Илгээгч:***Navigator.pushNamed(*

*context,*

*'/second',*

***arguments****: 'Hello from the first screen!',*

*);* **Хүлээн авагч:**   
*final String args = ModalRoute.of(context)!.settings.arguments as String;*

1. **Named Route ашиглан объект дамжуулах  
   Илгээгч:** *final product = Product(name: 'Laptop', price: 1000);  
   Navigator.pushNamed(context, '/product', arguments: product);* **Хүлээн авагч:**   
   *final Product product = ModalRoute.of(context)!.settings.arguments as Product;*

**Үр дүнг буцааж өмнөх дэлгэц рүү илгээх**

1. **Өмнөх дэлгэц:***final selectedPizza = await Navigator.of(context).push(*

*MaterialPageRoute(builder: (context) => PizzaSelectionScreen()),*

*);*

**Тайлбар:**

* **Navigator.of(context).push(...) — шинэ дэлгэц рүү шилжиж байна (PizzaSelectionScreen)**
* **await ашиглаж байгаа тул шинэ дэлгэц хаагдах хүртэл хүлээнэ**
* **Дэлгэц хаагдахад (pop) буцаасан утга selectedPizza хувьсагчид хадгалагдана**
* **setState() ашиглаж UI-г шинэчилнэ: Text('Selected Pizza: $\_selectedPizza')**

**Шинэ дэлгэц:***ListTile(*

*title: Text('Margherita'),*

*onTap: () {*

*Navigator.of(context).pop('Margherita');*

*},*

*),*

**Тайлбар:**

* **Хэрэглэгч Margherita дээр дарвал:**
* **Navigator.of(context).pop('Margherita') → утга буцаана**
* **pop() нь энэ дэлгэцийг хаана (буцах)**
* **push()-аар хүлээж байсан await дуусч, утга нь selectedPizza-д очно**

**Drawer цэс -** Хажуугийн цэсийг үүсгэн түүнийхээ 1 хэсэг дээр дарангууд бусад хуудас руу очдог байхаар мөн тодорхойлж болдог.

1. **Home цэс**

*ListTile(*

*title: Text('Home'),*

*onTap: () {*

*Navigator.pop(context); // Drawer-ийг хаах*

*},*

*),* ***энд үндсэн home хуудсан дээр байгаа учраас одоо байгаа context – ийг pop хийхэд үндсэн home хуудас үлдэнэ.***

1. **Contact цэс**

*ListTile(*

*title: Text('Contact'),*

*onTap: () {*

*Navigator.push(context, MaterialPageRoute(builder: (context) => ContactPage()));*

*},*

*),****Тусдаа contact хуудас байгаа учраас push хийн хуудас руу шилжин stack руу хийсэн***

1. **About цэс**

*ListTile(*

*title: Text('About'),*

*onTap: () {*

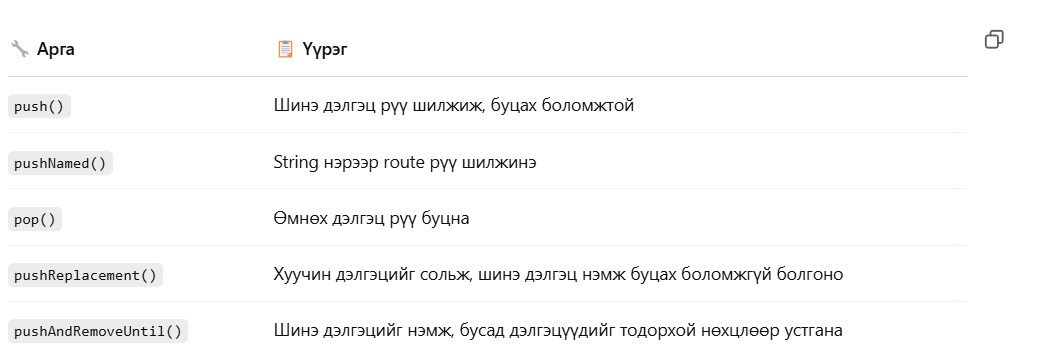
*Navigator.push(context, MaterialPageRoute(builder: (context) => AboutPage()));*

*},*

*),* ***Тусдаа about хуудас байгаа учраас push хийн хуудас руу шилжин stack руу хийсэн***

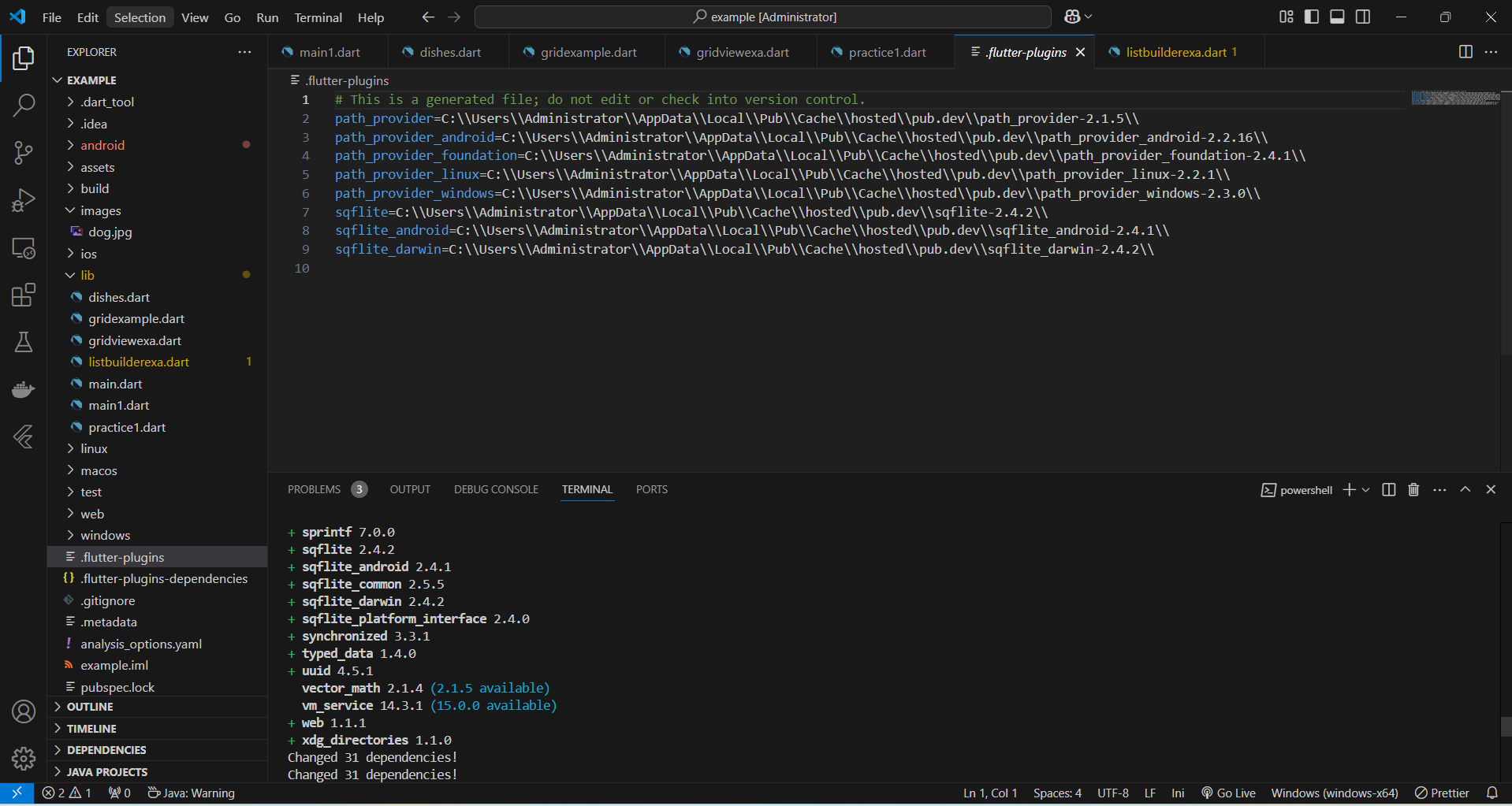
**Бусад дэлгэц рүү шилжих аргууд**

1. **Navigator.push() -** Шинэ дэлгэцийг navigation stack дээр нэмж, тухайн дэлгэц рүү шилжинэ.
2. **Navigator.pushNamed() -** Named route ашиглан дэлгэц рүү шилжих. Энэ нь route-уудыг MaterialApp-т бүртгэсэн байх шаардлагатай.
3. **Navigator.pop() -** Одоогийн дэлгэцийг хааж, өмнөх дэлгэц рүү буцна. Энэ нь утасны "Back" товчийн үйлдэлтэй ижил.
4. **Navigator.pushReplacement() -** Одоогийн дэлгэцийг устгаж, шинэ дэлгэцийг оронд нь тавина. Back дарсан ч хуучин дэлгэц рүү буцахгүй.
5. **Navigator.pushAndRemoveUntil() -** Шинэ дэлгэцийг нэмж, өмнөх дэлгэцүүдийг тодорхой нөхцөл хангах хүртэл устгана.  
   Жишээ нь, бүх дэлгэцийг устгаад login руу буцах. Энэ нь **logout** хийж байгаа тохиолдолд ашигтай.

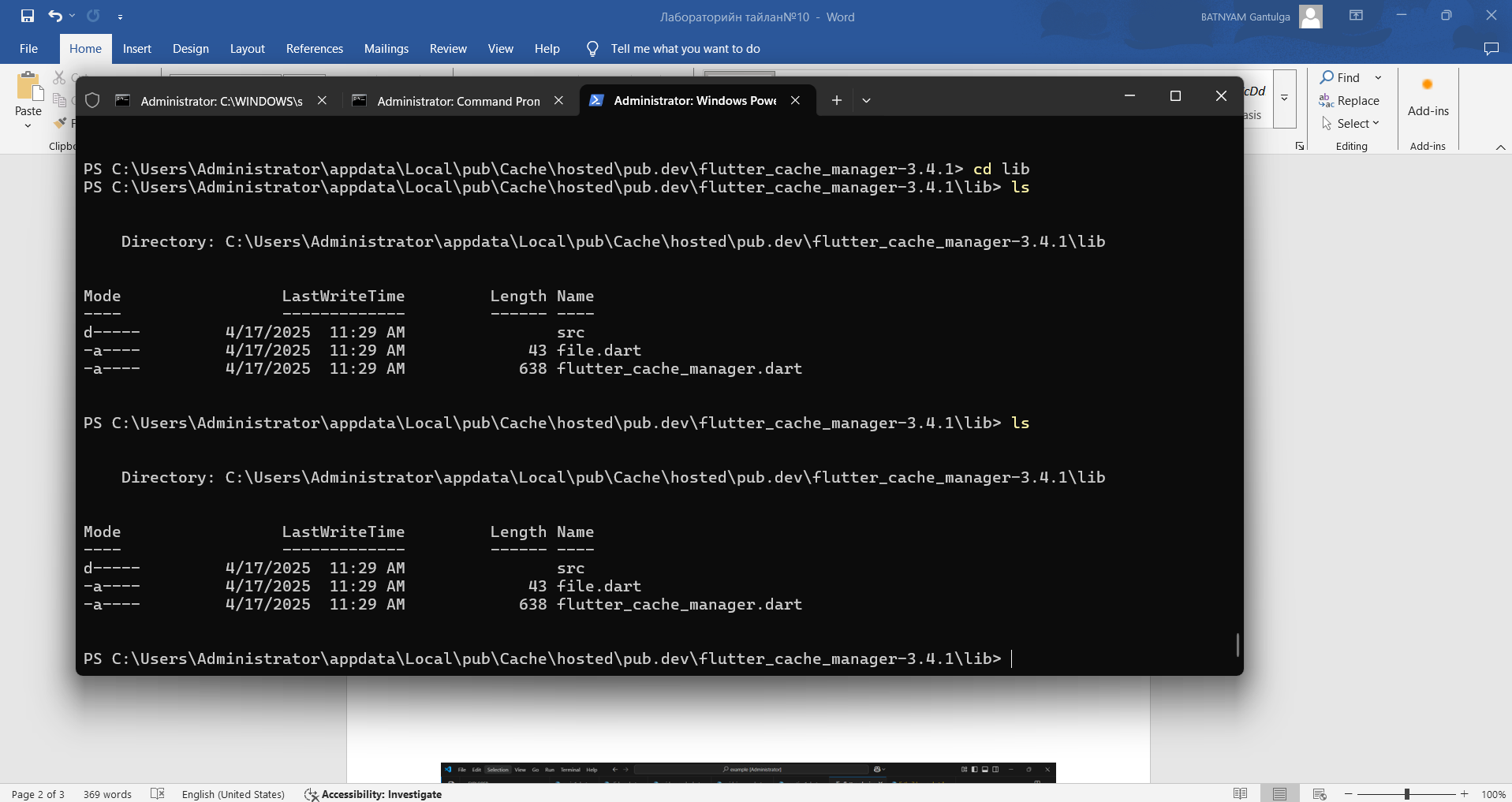
****

**TabBar болон TabBarView -** ашиглан таб хэлбэрийн (tabbed) интерфэйс хэрхэн хийх талаар хоёр жишээгээр тайлбарлаж байна. Энэ нь хэрэглэгчдэд дэлгэц хооронд хуруугаар гүйлгэх, эсвэл табууд дээр дарах замаар агуулга солих боломжийг олгодог. Эдгээрийг ашиглахын тулд **DefaultTabController**-оор бүхэлд нь боож өгөх шаардлагатай.Дэлгэцийн аль ч хэсэгт хэрэгжүүлж болдог.

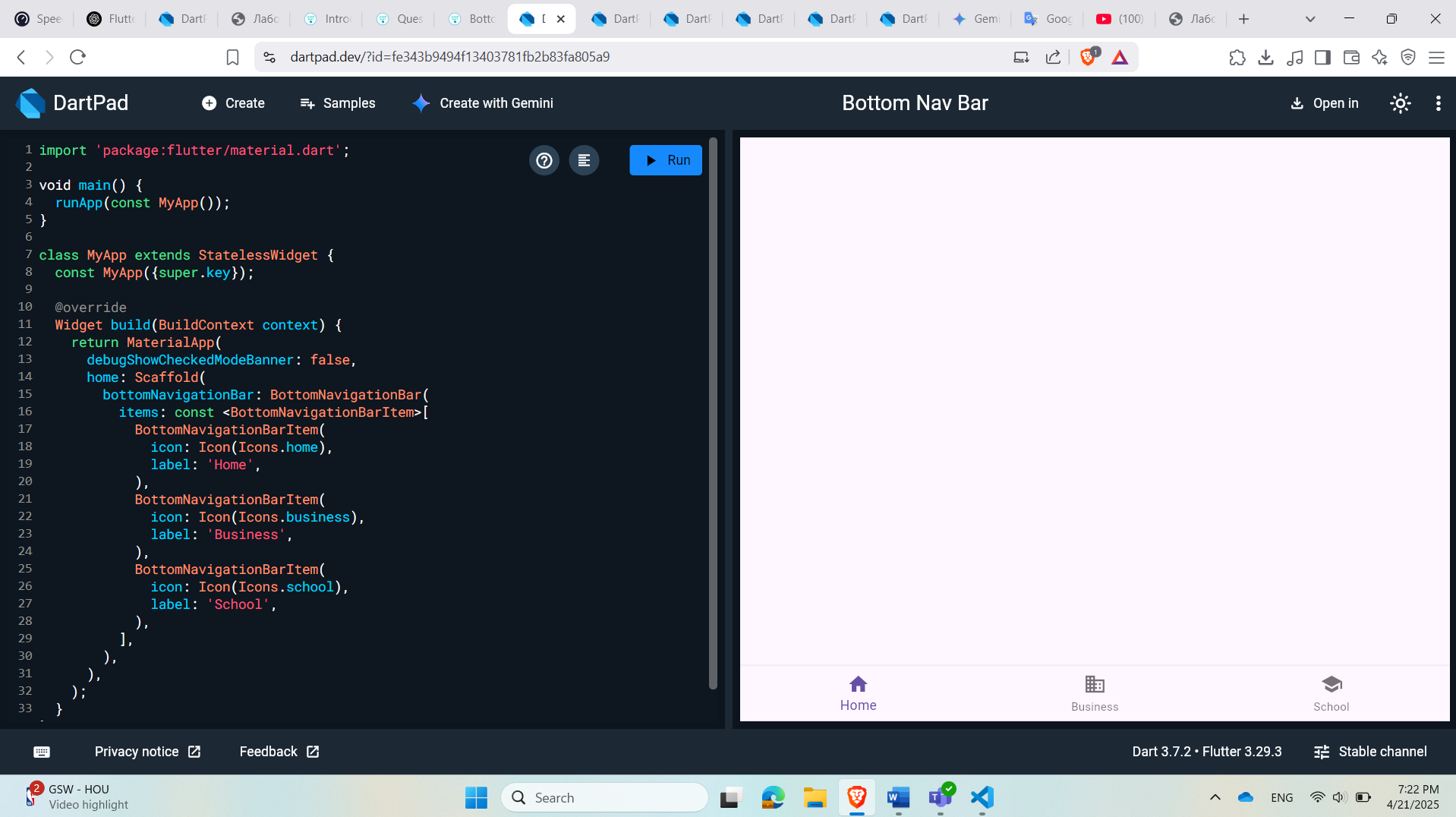
3.Хэрэгжүүлэлт



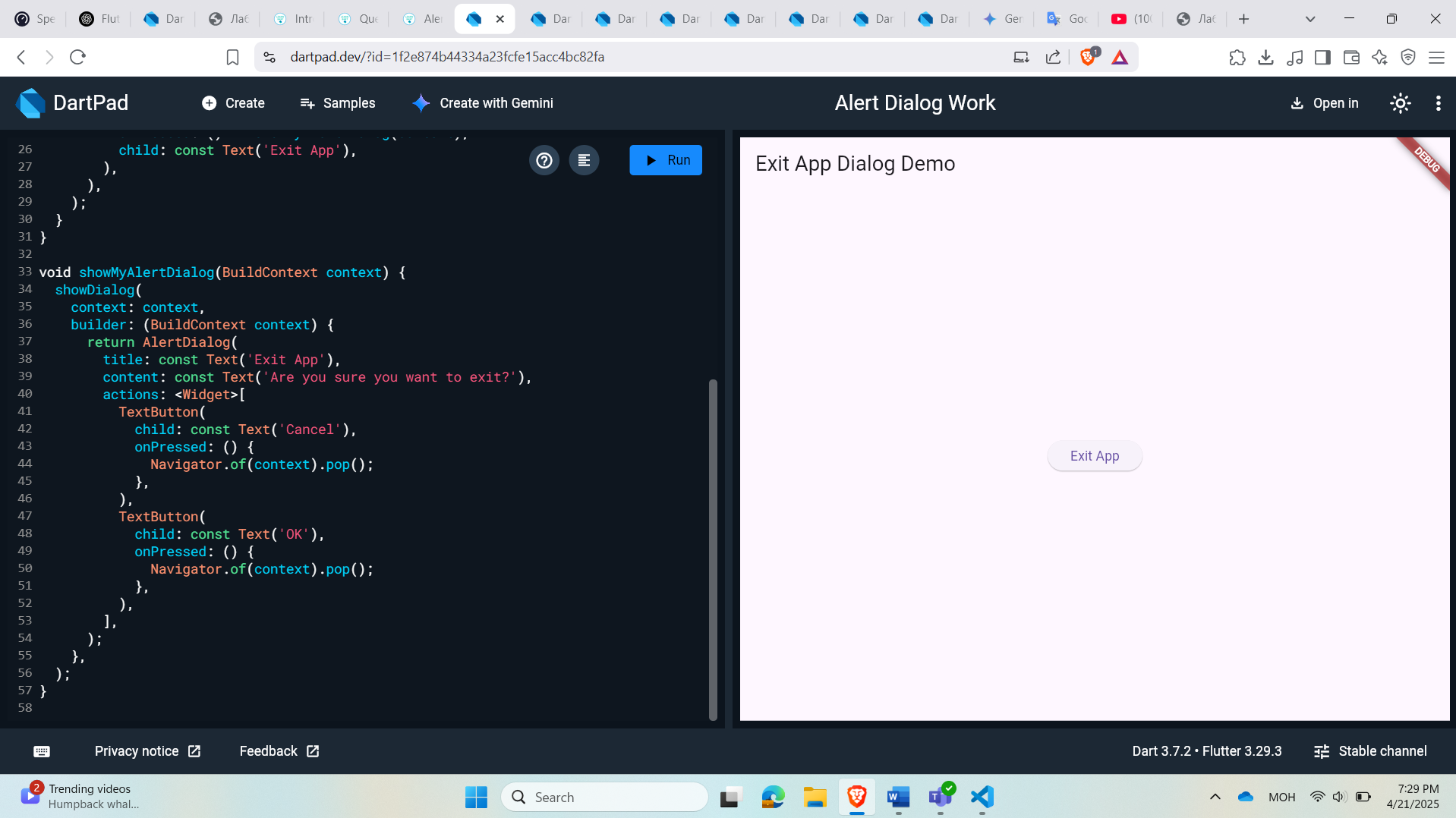
***Зураг1*** *cached\_network\_image амжилттай үүссэн*

******

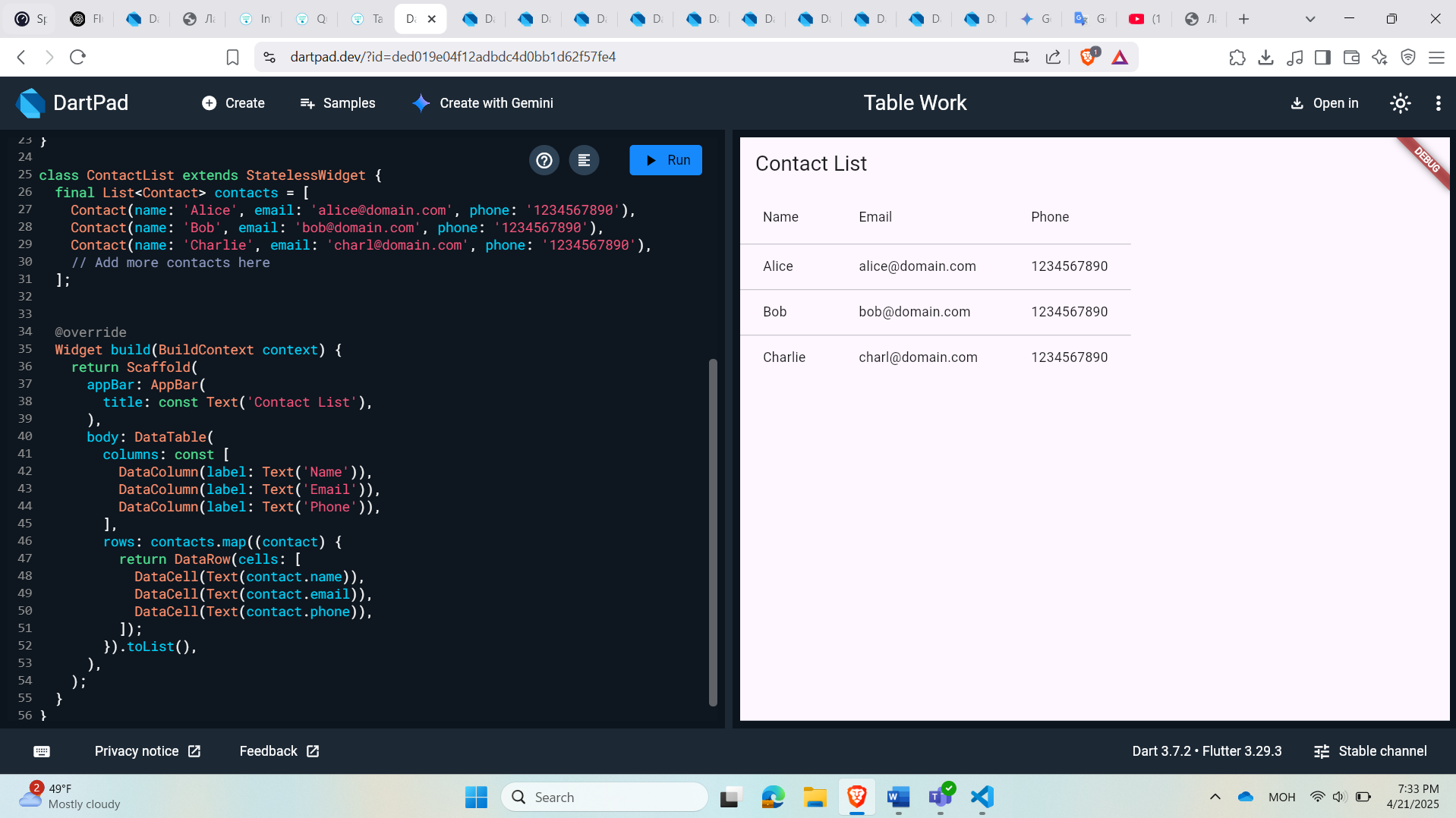
***Зураг 2*** *cache хийгдсэн зургууд дээрхи path – д хадгалагдаж байна*

******

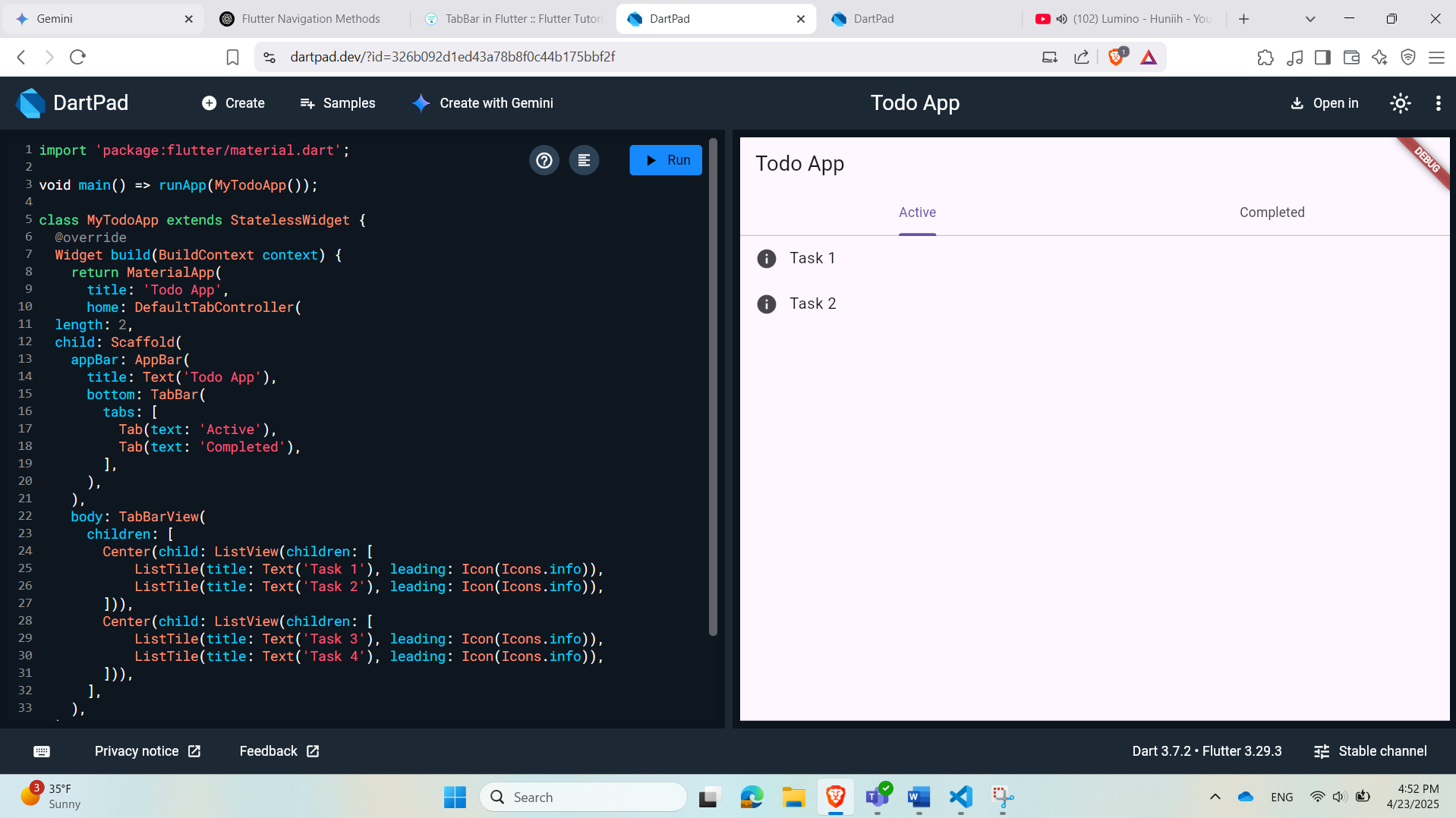
***Зураг3*** *Доод хэсэгт байрлах BottomNavigationBar ашигласан байдал*

**

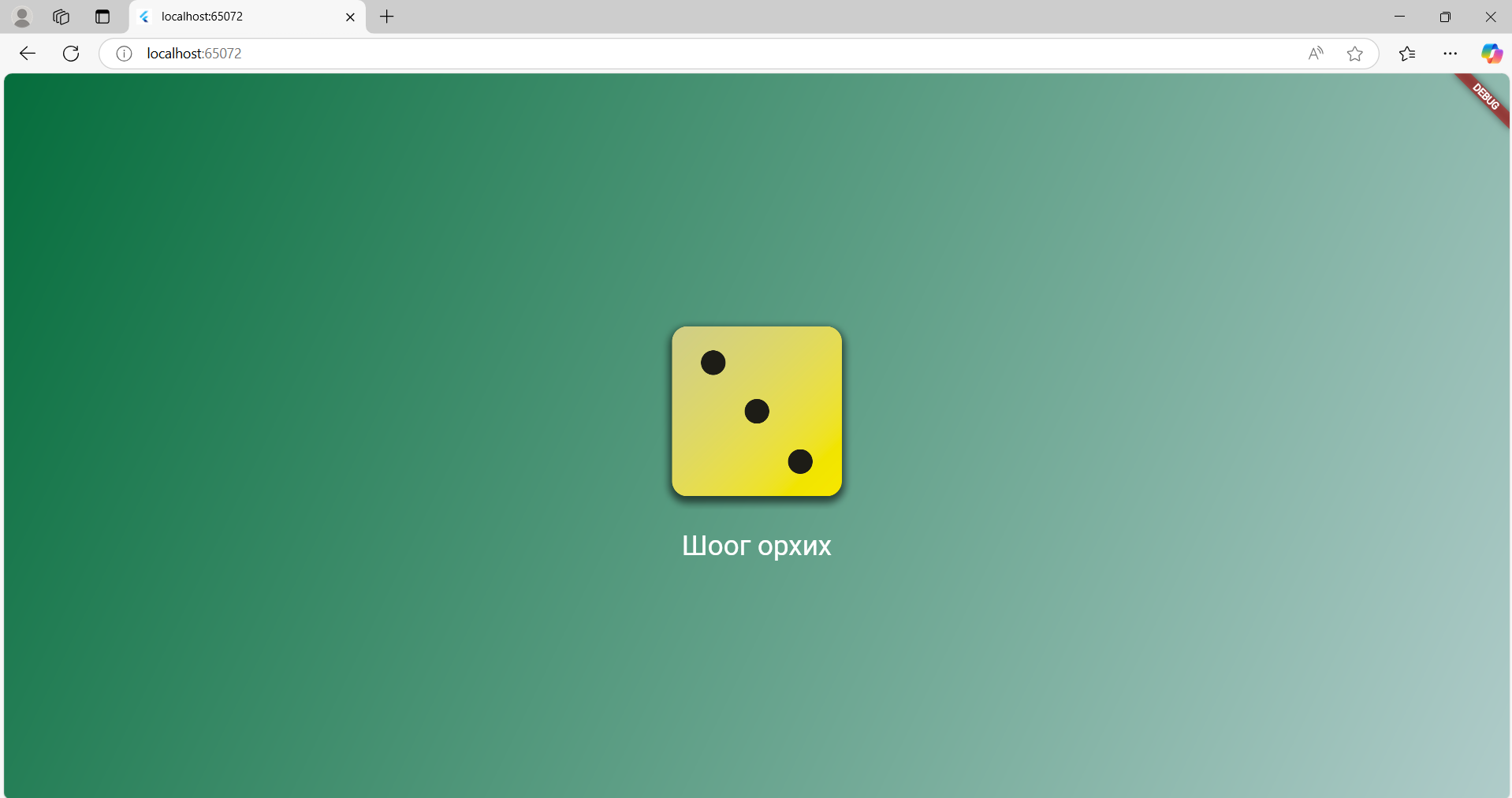
***Зураг4*** *AlertDialog хэрэгжүүлсэн байдал*

**

***Зураг5*** *Хүснэгтийг хэрэгжүүлсэн байдал*



***Зураг6*** *Дэлгэцийг tab болгон хуваан тэдгээр хоорондоо шилжих дасгал*

**

***Зураг7*** *Эхний даалгарыг хийж гүйцэтгэсэн*

4.Дүгнэлт

Энэ лабораторийн ажлаар flutter – ийн үндсэн ойлголтууд болох asset буюу дотоод болон онлайн өгөгдөл боловсруулалт мөн дэлгэцийн мэдэгдлүүд, дэлгэцийн navigation bar, alertDialog, button, navigation – ийг туршиж хэрэгжүүлж өгөгдсөн даалгавруудыг хийж гүйцэтгэлээ.

5. Ашигласан материал

(https://flutter-tutorial.net/useful-widgets/, n.d.)

(https://flutter-tutorial.net/list-and-grid/, n.d.)

(https://flutter-tutorial.net/working-with-assets/, n.d.)

(https://flutter-tutorial.net/buttons-in-flutter/, n.d.)

(https://flutter-tutorial.net/forms-in-flutter/, n.d.)

(https://flutter-tutorial.net/navigation-in-flutter/, n.d.)