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 даалгаврууды 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(

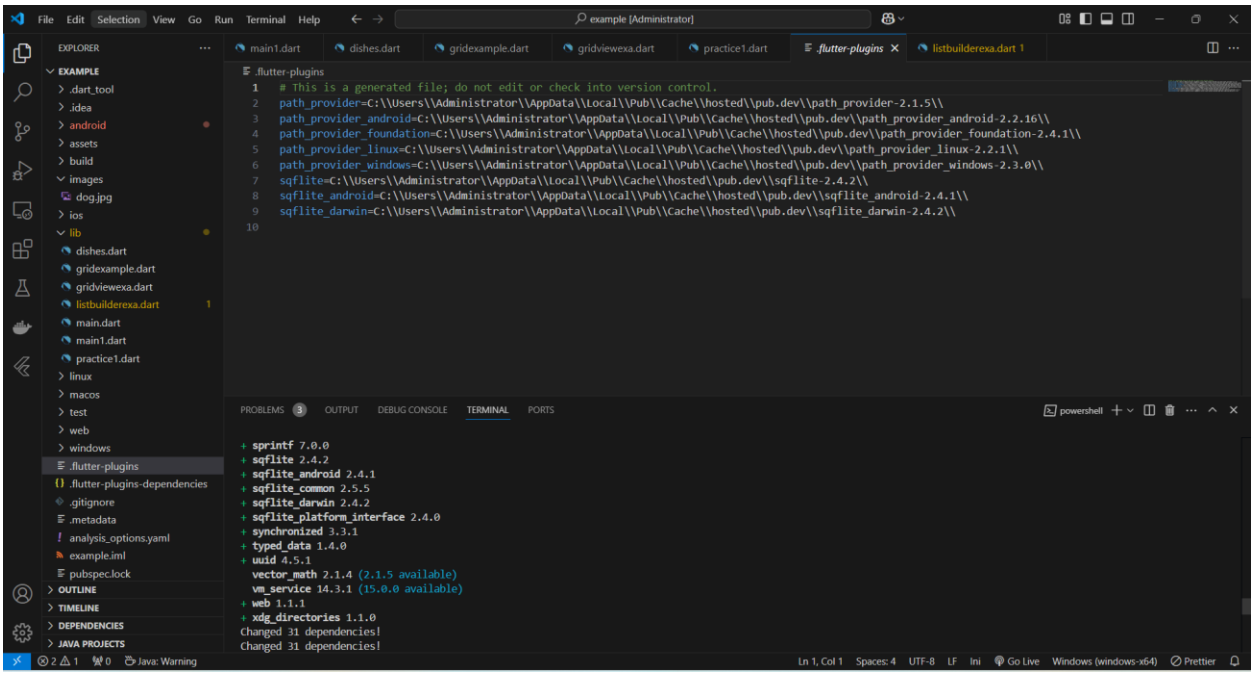
```
  content: Text('Message deleted'),
  backgroundColor: Colors.blue,
  duration: Duration(seconds: 10),//нийт үргэлжлэх хугацаа
  action: SnackBarAction(
   label: 'UNDO',
   onPressed: () {
    // Perform some action
   },
  ),//button хэрэгжүүлэх үед
 ),
);
```

Жишээ нь:

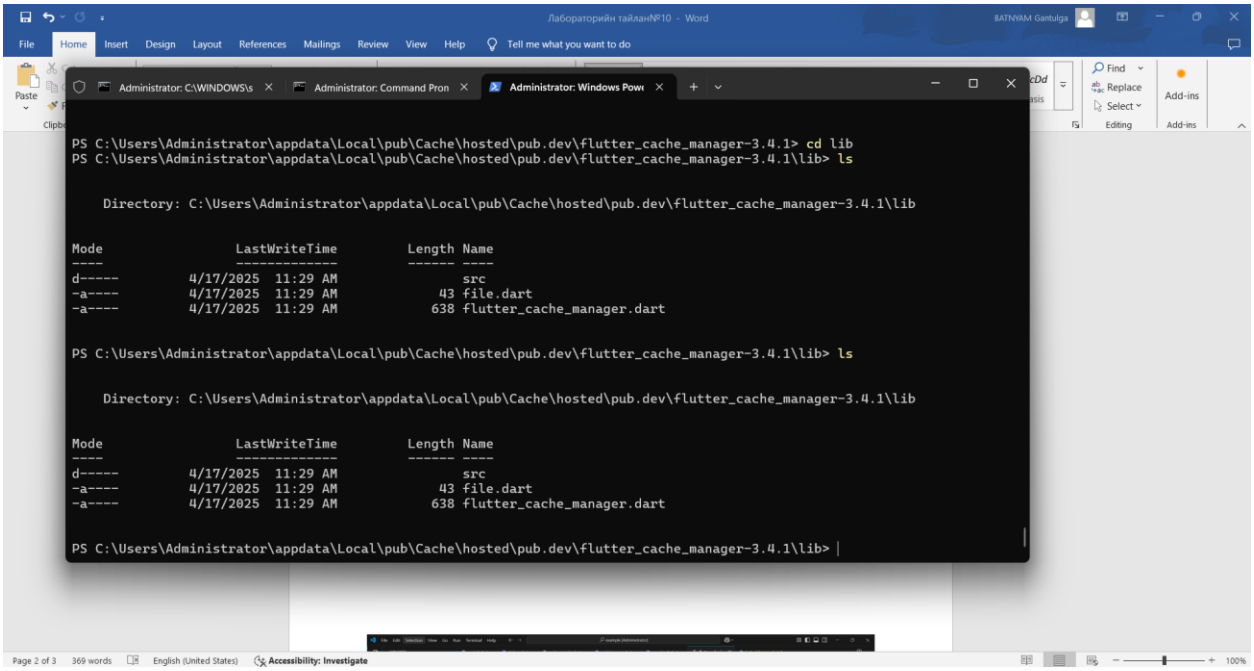
```
ScaffoldMessenger.of(context).showSnackBar(
        const SnackBar(
          content: Text("Account created successfully"),
          duration: Duration(seconds:5),
          backgroundColor:Colors.green,
        ),
```

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.

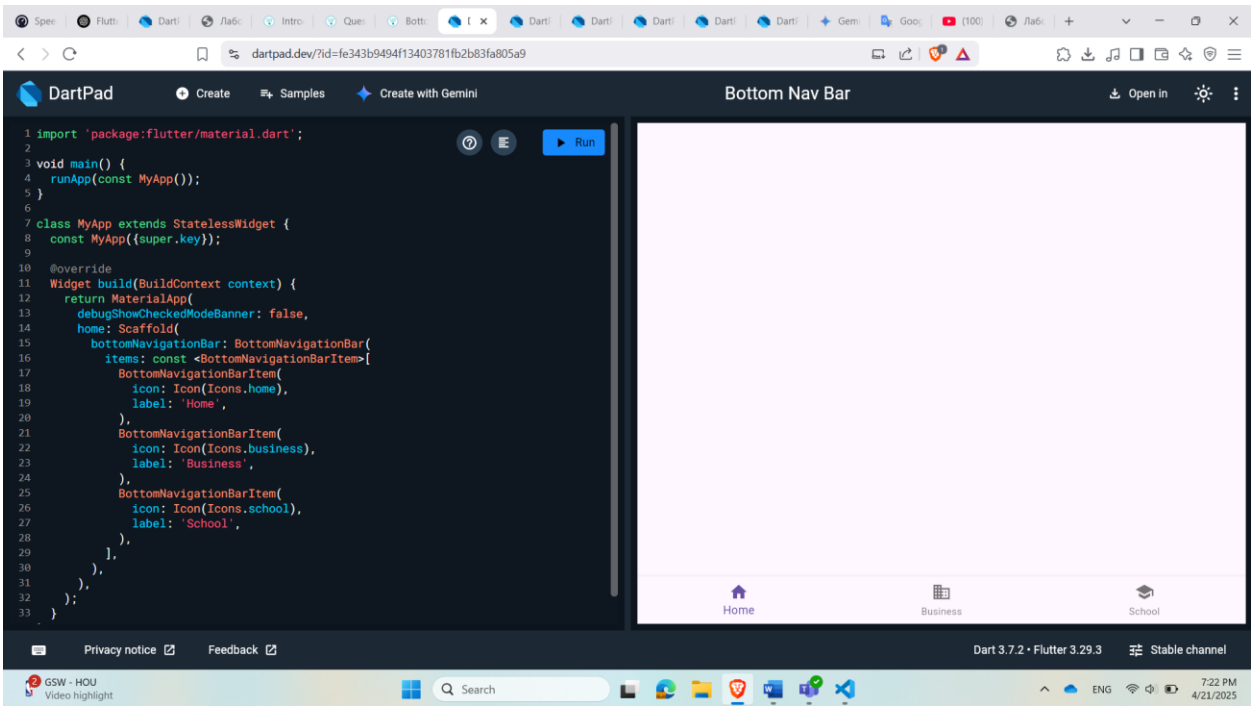
3.Хэрэгжүүлэлт



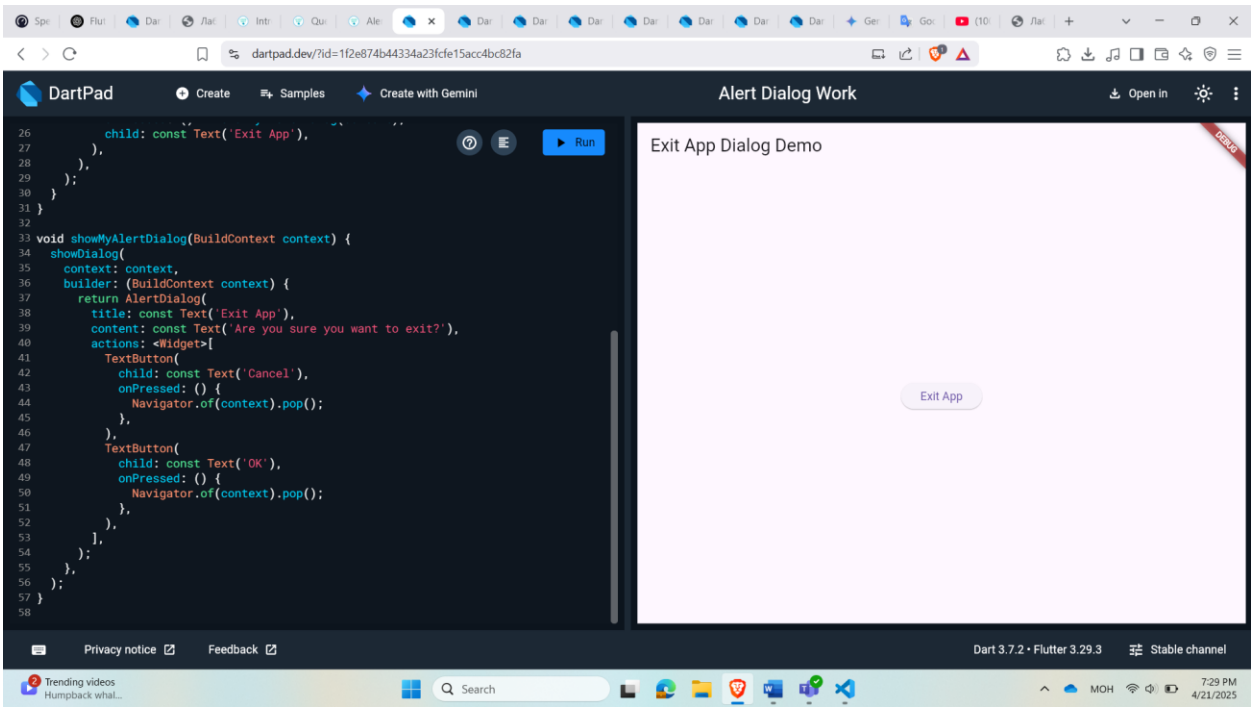
Зураг1 *cached_network_image амжилттай үүссэн*



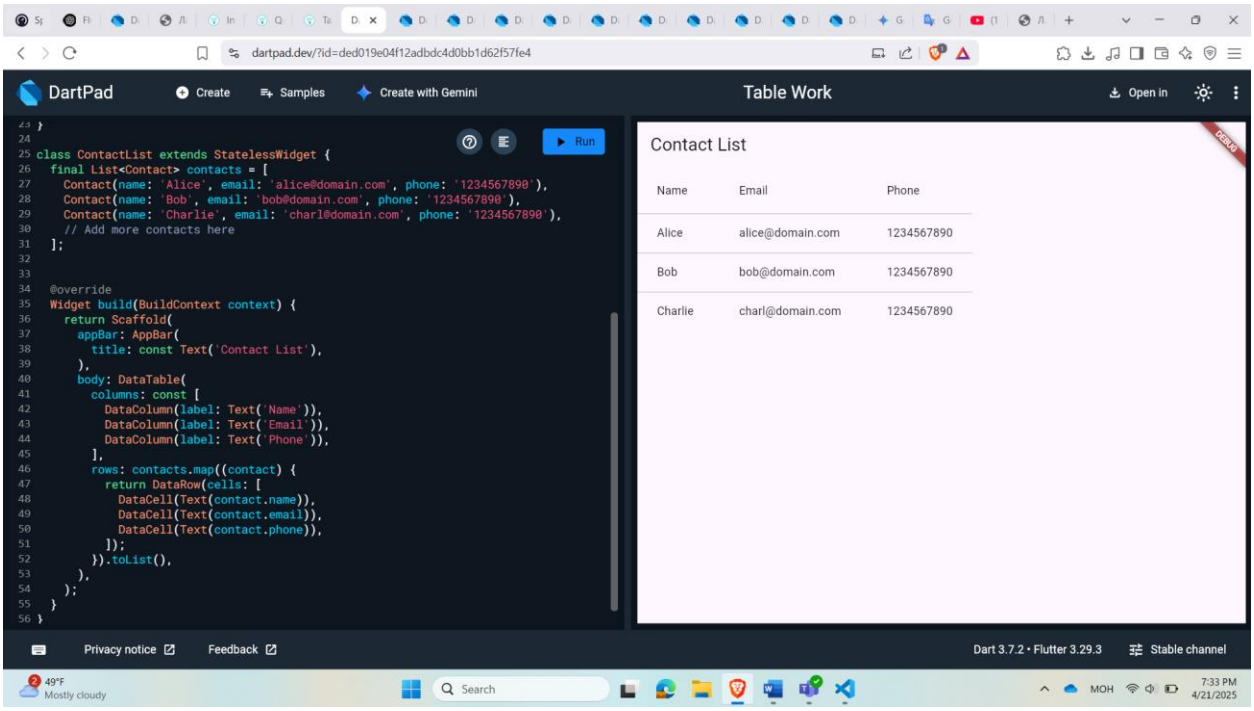
Зураг 2 *cache хийгдсэн зургууд дээрхи path – д хадгалагдаж байна*



Зураг3 *Доод хэсэгт байрлах BottomNavigationBar ашигласан байдал*



Зураг4 *AlertDialog хэрэгжүүлсэн байдал*



Зураг5 *Хүснэгтийг хэрэгжүүлсэн байдал*

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 даалгаруудыг 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.)