

PHẦN 1: NGÔN NGỮ DART

Mục tiêu: Làm quen với ngôn ngữ Dart, cú pháp, kiểu dữ liệu nullable, lập trình bất đồng bộ.

Công cụ: IDE Online <https://dartpad.dev/?>

Bài tập 1: Viết chương trình Hello World để kiểm tra IDE

Bài tập 2:

- Khai báo lớp SinhVien bao gồm các thuộc tính:

+ id, ten: bắt buộc.

+ ngaySinh, que_Quan: Không bắt buộc

- Yêu cầu:

1. Viết hàm khởi tạo SinhVien cho các thuộc tính đã khai báo (Sử dụng named parameter).

2. Ghi đè hàm toString sẵn có để trả về thông tin của một sinh viên.

3. Viết lớp QL_SinhVien để quản lý một danh sách sinh viên với các phương thức add, edit, delete với đối số là một SinhVien.

Bài tập 3:

1. Viết hàm để kiểm tra một danh sách có chứa các phần tử giống nhau hay không (để đơn giản, có thể sử dụng danh sách số nguyên).

2. Nghịch lý Ngày sinh nhật (Birthday Paradox) được phát biểu như sau: “Trong một nhóm có 23 người bất kỳ, xác suất để có hai người có cùng một ngày sinh nhật là không ít hơn $\frac{1}{2}$ ”.

Hãy kiểm tra lại giả thuyết trên bằng thực nghiệm.

Bài tập 4:

1. Sử dụng phương thức Map của List để chuyển một danh sách các chuỗi thành danh sách chiều dài của các chuỗi đó.

2. Sử dụng phương thức forEach của List để in ra bình phương các phần tử của một danh sách.

Bài tập 5:

Viết hàm nhận vào một đối số, kiểm tra:

- nếu đối số là chuỗi thì in ra chuỗi viết hoa tương ứng.

- Nếu đối số là số nguyên thì in ra các ước số của số nguyên đó.

- Nếu đối số là số thực (float) thì in ra phần nguyên của số thực đó

Bài tập 6:

1. Viết hàm yêu nhận hai tham số: Tên người dùng và năm sinh của họ. Cho biết họ phải sống bao nhiêu năm nữa để được 100 tuổi.
2. Viết hàm nhận vào một danh sách li, trả về danh sách các số dư của các phần tử của danh sách li trong phép chia cho 5.
3. Viết hàm nhận vào hai danh sách và trả về danh sách các phần tử xuất hiện ở cả hai danh sách nhận vào.

Bài tập 7:

Viết chương trình tạo ra một danh sách ngẫu nhiên 10 phần tử từ 0 đến 100.

- In danh sách 10 phần tử trên
- Tạo và in ra danh sách bao gồm phần tử đầu tiên và phần tử cuối cùng của danh sách trên.

Bài tập 8:

1. Viết hàm đảo ngược một danh sách.
2. Viết hàm reverseString nhận vào một chuỗi và trả về chuỗi đảo ngược.

Bài tập 9: Future Type

1. Viết hàm trả lateNum về một số nguyên ngẫu nhiên sau khi thực hiện lời hàm 1s.
2. Kiểm tra số trả về bởi hàm lateNum là chẵn hay lẻ (in kết quả ra màn hình).

Hướng dẫn:

1.

- Google keyword: “Generate random numbers in Dart”
- Hàm lateNum có thể viết như sau:

```
import 'dart:math';  
Future<int> lateNum(){  
  var one = Duration(seconds: 1);  
  return Future.delayed(one).then(  
    (value) => Random().nextInt(1000),  
  );  
}
```

Bài tập 10: Stream, async, async*, await

- Viết hàm trả về một Stream<int> gồm 10 số nguyên.
- Viết hàm tính tổng 10 số nguyên của Stream

Bài tập 11: Stream, async, await, StreamController

- Viết lớp MyStream<T>

```
class MyStream<T>{
    StreamController<T> _streamController = StreamController();
    Stream<T> get stream => _streamController.stream;
    void addEvent(T event){
        _streamController.sink.add(event);
    }
    void dispose(){
        _streamController.close();
    }
}
```

- Tính tổng 10 số nguyên (ứng với 10 Event) do một đối tượng thuộc lớp MyStream sinh ra.

PHẦN 2: FLUTTER CƠ BẢN

Bài tập 1:

Sinh viên chú ý: Cần thực hiện bài tập này ở nhà, thời gian cài đặt chừng 1 buổi nếu mạng tốt, không thể thực hiện trên lớp.

Bài tập này rất quan trọng và nó ảnh hưởng đến toàn bộ các bài tập khác trong môn học, đề nghị các bạn sinh viên nghiêm túc thực hiện đúng theo thời gian qui định.

Nếu máy tính cấu hình yếu, ram dưới 8GB có thể không thực hiện được việc cài đặt này.

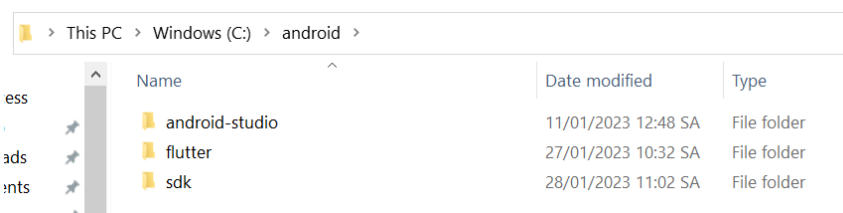
Cài đặt môi trường phát triển cho ứng dụng Flutter

- Cài đặt Android Studio
- Cài đặt các android SDK
- Cài đặt Plugin: Flutter, Dart
- Cài đặt máy ảo Android hoặc máy ảo Blue Stack (khuyến dùng)
- Test môi trường cài đặt: Tạo project Flutter với ứng dụng mặc định: Counter.
- Test ứng dụng trên máy ảo, điện thoại thật chạy Android.

Hướng dẫn:

1. Tải tệp cài đặt tại: <https://developer.android.com/studio>, cuộn xuống dưới, chọn tệp cài đặt có đuôi .zip.

2. Giải nén tệp cài đặt vào thư mục C:\android,

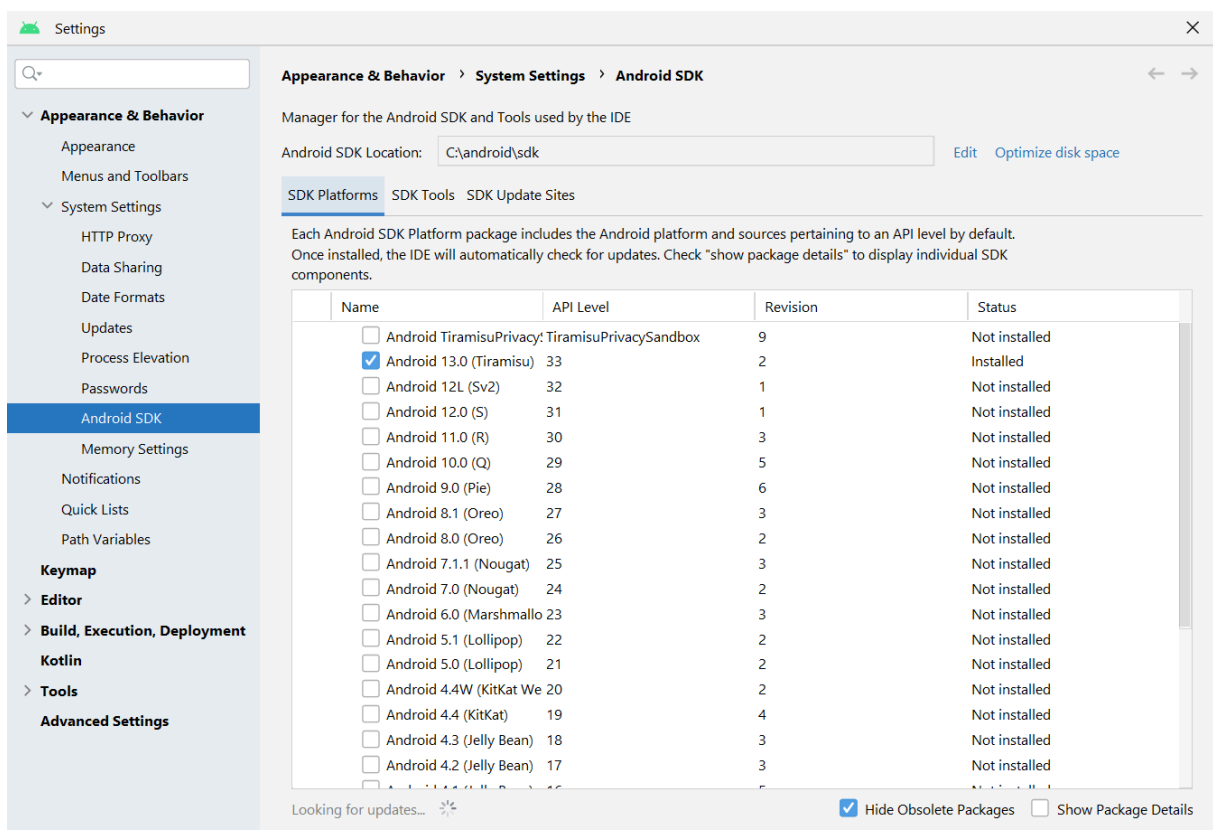
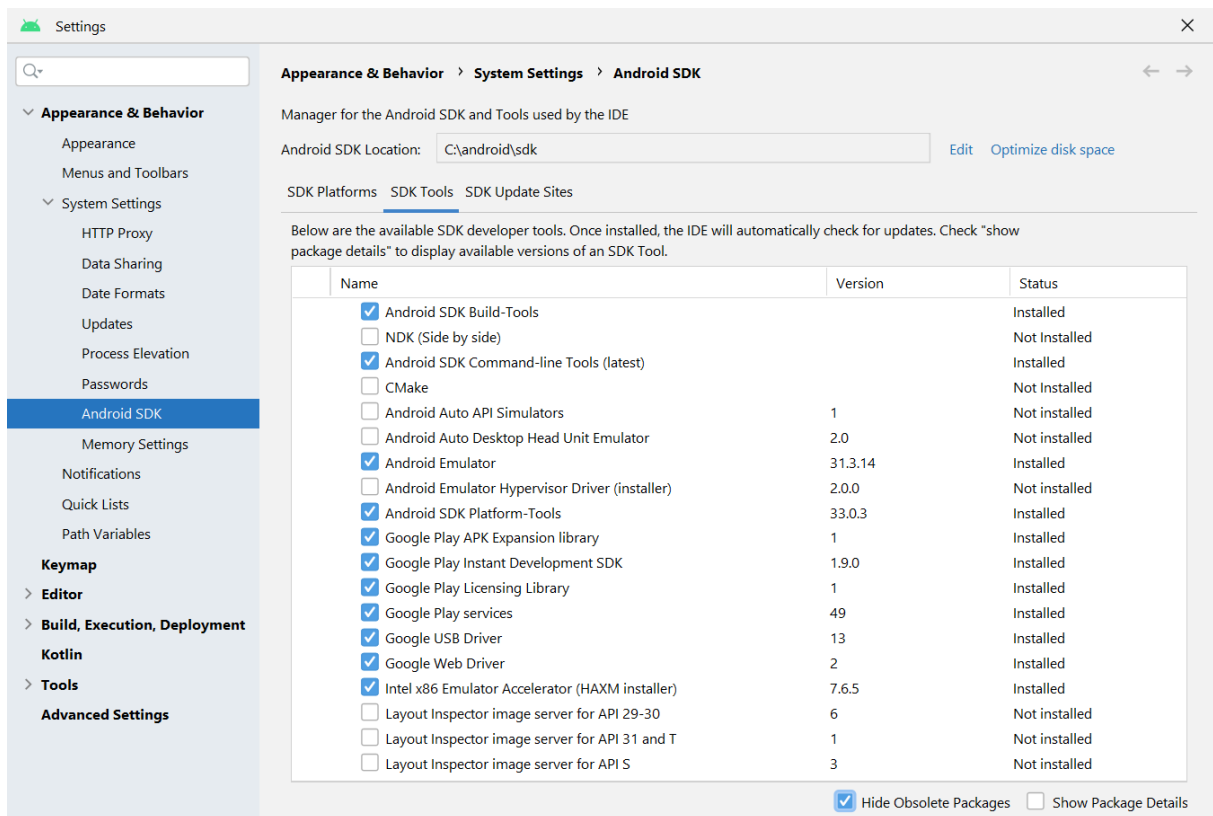


Name	Date modified	Type
android-studio	11/01/2023 12:48 SA	File folder
flutter	27/01/2023 10:32 SA	File folder
sdk	28/01/2023 11:02 SA	File folder

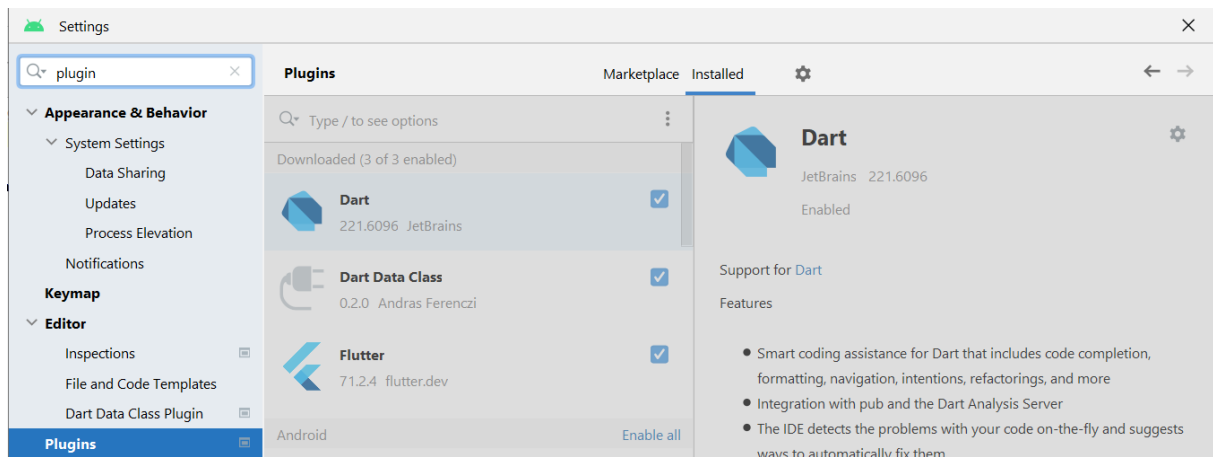
3. Chép nội dung của thư mục C:\android\android-studio\jbr vào thư mục C:\android\android-studio\jre.

4. Tải Flutter SDK tại địa chỉ: <https://docs.flutter.dev/development/tools/sdk/releases>. Giải nén vào thư mục C:\android như hình trên.

5. Chạy tệp tin C:\android\android-studio\bin\studio64.exe. Cài đặt các thành phần sau như hình dưới đây, nhớ chọn lại thư mục cài đặt là C:\android\sdk. Các bạn có thể bỏ qua trong lần chạy đầu tiên và chọn cài đặt ở lần chạy tiếp theo trong mục File Setting. Ở lần đầu tiên, cần phải cài Plugin Flutter để có thể tạo dự án Flutter (Bước 7).

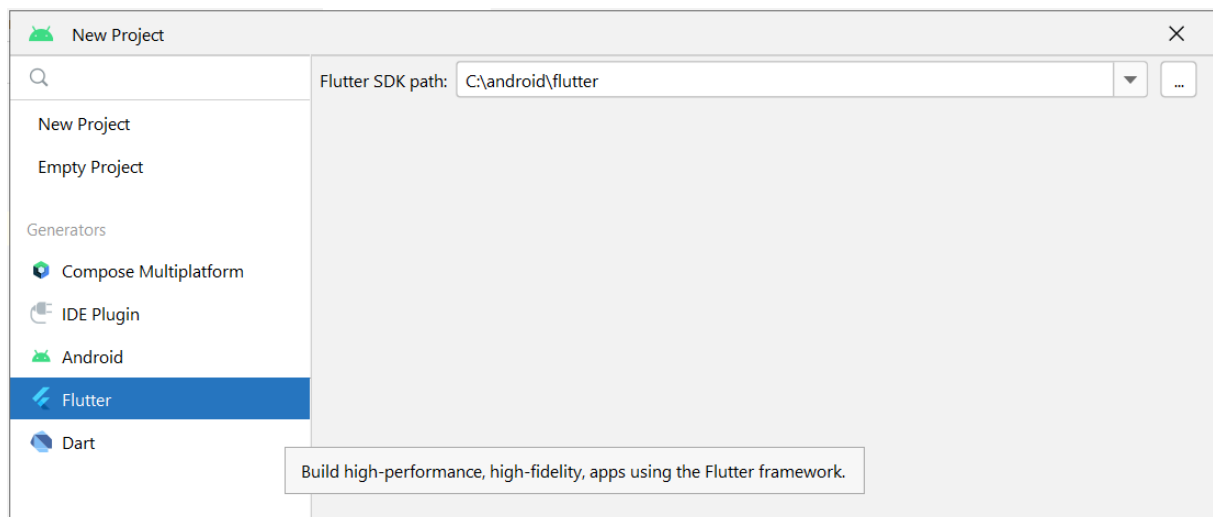


6. Phần plugin: Cài đặt các Plugin sau:



Bước 7: Tạo dự án Flutter để test (Nhớ: khi tạo, chạy dự án Flutter đều phải kết nối internet)

Phần Project name: chỉ được sử dụng ký tự viết thường không dấu, không để khoảng trắng.



New Project

Project name:

Project location: ...

Description:

Project type:

Organization:

Android language: ☒ Java ☐ Kotlin

iOS language: ☐ Objective-C ☒ Swift

Platforms: ☒ Android ☒ iOS ☐ Linux ☐ MacOS ☐ Web ☐ Windows

When created, the new project will run on the selected platforms (others can be added later).

☐ Create project offline

▼ More Settings

Module name:

Content root: ...

Module file location: ...

Project format:

Cài đặt các mục như bước 6, nếu chưa cài đặt

8. Cài đặt máy ảo test ứng dụng, khuyên các SV nên cài đặt BlueStack để chạy ứng dụng được tốt hơn. Máy ảo của Android Studio khởi động nhanh nhưng chiếm dụng nhiều RAM và CPU, nếu máy yếu sẽ làm chậm cả hệ thống. Thiết lập máy ảo như sau (Nhớ chọn ngôn ngữ Tiếng Anh):

Phần Display, nhớ chọn theo một thông số điện thoại thật, trong hướng dẫn là thông số của điện thoại SS A33.

Settings



Performance

Display

Graphics

Devices

Gamepad

Preferences

Phone

Shortcuts

User data

Advanced

About

CPU allocation

Medium (2 Cores)



Memory allocation

Medium (2 GB)



Performance mode

Balanced



Frame rate : 30

1



60

Recommended FPS

Play at 60 FPS for smooth gameplay. If you're running multiple instances, we recommend selecting 20 FPS.

Enable high frame rate




Enable VSync (to prevent screen tearing)

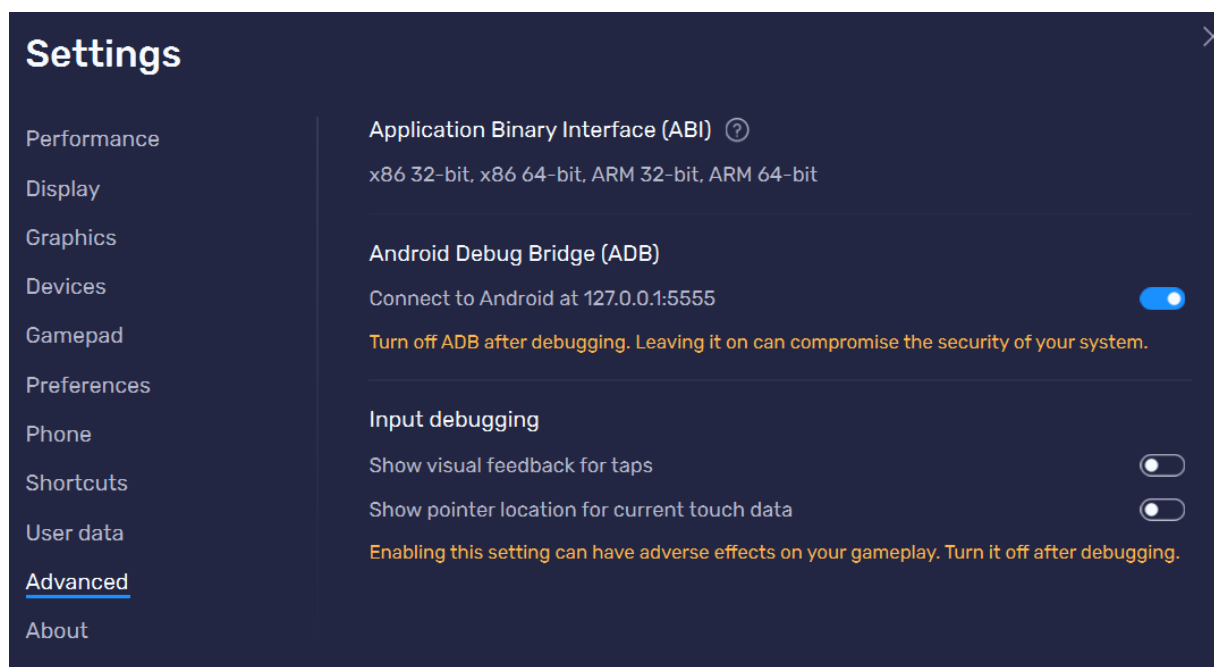
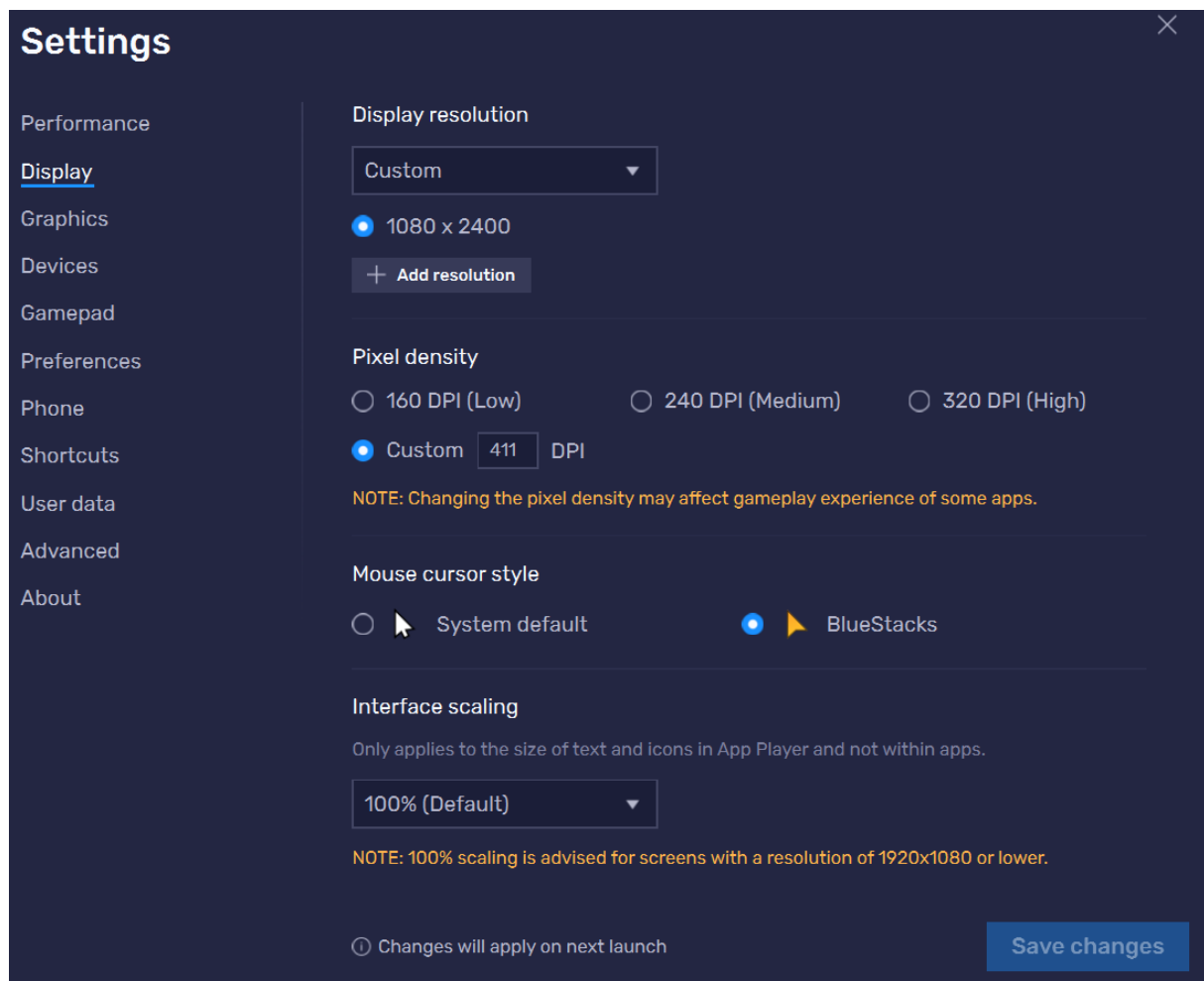


Display FPS during gameplay



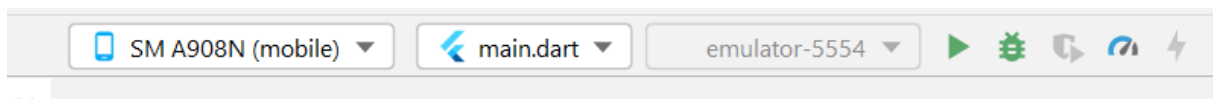
 Some changes will apply on next launch

Save changes



9. Test thử môi trường phát triển ứng dụng:

- Khởi động máy ảo Blue Stack, mở project hello_world.
- Trong cửa sổ Terminal lần lượt chạy 2 câu lệnh sau: `adb kill-server`, `adb start-server`. Sau khi chạy hai câu lệnh này sẽ thấy máy ảo được nhận trong Android Studio.



- Kiểm tra môi trường phát triển ứng dụng: Trong cửa sổ Terminal chạy câu lệnh: “flutter doctor”. Nếu kết quả như dưới đây thì môi trường phát triển ứng dụng đã được cài đặt thành công. (2 vấn đề cảnh báo: Chrome, và Visual studio chưa được cài đặt trên máy tính không ảnh hưởng đến việc phát triển ứng dụng trên mobile)

```
[V] Windows Version (Installed version of Windows is version 10 or higher)
[V] Windows Version (Installed version of Windows is version 10 or higher)
[V] Android toolchain - develop for Android devices (Android SDK version 33.0.1)
[X] Chrome - develop for the web (Cannot find Chrome executable at .\Google\Chrome\Application\chrome.exe)
    ! Cannot find Chrome. Try setting CHROME_EXECUTABLE to a Chrome executable.
[X] Visual Studio - develop for Windows
    X Visual Studio not installed; this is necessary for Windows development.
      Download at https://visualstudio.microsoft.com/downloads/.
      Please install the "Desktop development with C++" workload, including all of its default components
[V] Android Studio (version 2022.1)
[V] Connected device (3 available)
[V] HTTP Host Availability

! Doctor found issues in 2 categories.
```

- Đưa ứng dụng lên máy ảo bằng cách click vào nút play (màu xanh). Chú ý máy tính cần phải kết nối với internet vì khi triển khai ứng dụng lần đầu tiên IDE sẽ tải và cài đặt một số công cụ/thư viện cần thiết.

Bài tập 2: Ứng dụng hiển thị Profile cá nhân

Hiển thị các thông tin các nhân đơn giản như: Họ tên, Ngày sinh, Quê quán, sở thích, giới tính trên một trang của ứng dụng Flutter.



Hướng dẫn:

- Trang ứng dụng có thể là một StatelessWidget hoặc là một StatefulWidget. Tuy nhiên, trang ứng dụng nên được thiết kế là một StatefulWidget để có thể cập nhật trạng thái của ứng dụng như yêu cầu trong bài tập 3.

- Vẽ Widget Tree của ứng dụng

- Gợi ý:

- Thiết kế giao diện theo Column Layout.
- Ảnh đại diện: Lưu ảnh trong assets của ứng dụng.
- Để hiển thị ảnh có kích thước như mong muốn và căn giữa màn hình, bọc Image trong Container, sau đó tiếp tục bọc Container trong một Center.

```
Center(
  child: Container(
    height: size*2/3,width: size,
    child: Image.asset(myProfile.imageAssest)),
),
```

- Muốn kích thước ảnh hiển thị phụ thuộc vào kích thước thiết bị, sử dụng MediaQuery.of(context) để truy vấn kích thước của thiết bị:

`MediaQuery.of(context).size.width`

- Sử dụng các `SizedBox` widget để tạo khoảng cách giữa các Widget khác trong ứng dụng. Thuộc tính *width* cho phép thiết lập khoảng cách theo chiều ngang, thuộc tính *height* cho phép thiết lập khoảng cách theo chiều dọc
- Sử dụng `Text` để hiển thị nội dung văn bản.

Bài tập 3:

Mục đích:

- Thiết kế giao diện của ứng dụng Flutter
- Thực hành các thành phần của Scaffold: `AppBar`, `BottomAppBar`, `FloatingActionButton`, `Drawer`, `SnackBar`, `BottomSheet`...

Thiết kế Ứng dụng với giao diện như hình dưới đây.

Hướng dẫn:

1. Sinh viên tìm hiểu các thành phần của `Drawer`:

- children: Gồm `DrawerHeader` và các item (có thể sử dụng `ListTile` cho mỗi item)
- `DrawerHeader`: sử dụng `UserAccountsDrawerHeader`
- Ảnh trong `DrawerHeader`: thuộc tính `currentAccountPicture` của `UserAccountsDrawerHeader` là một `CircleAvatar`

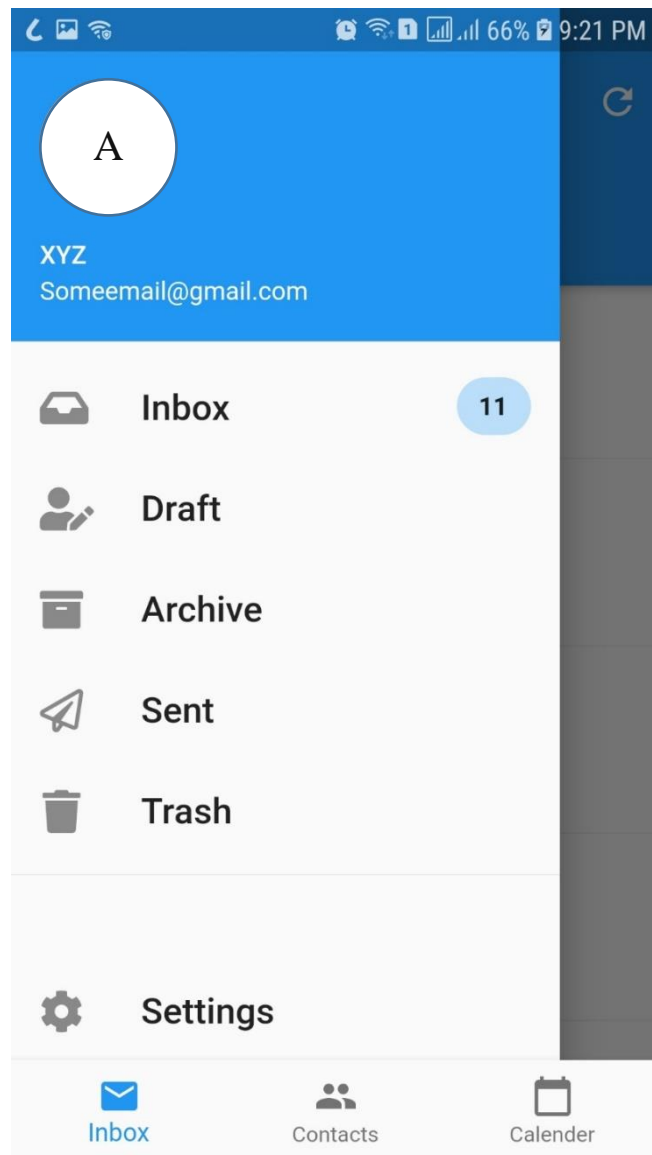
2. Tìm hiểu `BottomNavigationBar`: Các thành phần của `BottomNavigationBar`

- items: Danh sách các `BottomNavigationBarItem`
- Mỗi item có một index, item đầu tiên có index 0.
- Sự kiện `onTap`: Sự kiện khi người dùng click vào một item. Có thể sử dụng cấu trúc `switch...case` dựa vào các giá trị `currentIndex` được thiết lập để xử lý các sự kiện.

```

return Scaffold(
  bottomNavigationBar: BottomNavigationBar(
    type: BottomNavigationBarType.fixed,
    currentIndex: _currentIndex,
    backgroundColor: colorScheme.surface,
    selectedItemColor: colorScheme.onSurface,
    unselectedItemColor: colorScheme.onSurface.withOpacity(.60),
    selectedLabelStyle: textTheme.caption,
    unselectedLabelStyle: textTheme.caption,
    onTap: (value) {
      // Respond to item press.
      setState(() => _currentIndex = value);
    },
    items: [
      BottomNavigationBarItem(
        title: Text('Favorites'),
        icon: Icon(Icons.favorite),
      ),
      BottomNavigationBarItem(
        title: Text('Music'),
        icon: Icon(Icons.music_note),
      ),
    ],
  ),
)

```



Bài tập 4: Phát triển dựa trên bài tập 2

Thêm vào nút chỉnh sửa để mở một trang mới cho phép chỉnh sửa các thông tin như họ tên, quê quán, ngày sinh.

Hướng dẫn:

- Cả hai trang của ứng dụng đều là các StatefulWidget.
- Định nghĩa một class Profile cho đối tượng là state trong widget đầu tiên

```
class Profile{  
  String hoTen;  
  String queQuan;  
  DateTime ngaySinh;  
  String soThich;  
  String imageAssest;  
  
  Profile({this.hoTen, this.queQuan, this.ngaySinh, this.soThich,
```

```

        this.imageAssest});
    }

```

- Khởi tạo trạng thái (state) cho trang thứ nhất:

```

@override
void initState() {
    // TODO: implement initState
    super.initState();
    myProfile = Profile(
        hoTen: "Trần Thành Công",
        ngaySinh: DateTime(2000,9,11),
        imageAssest: 'assets/bai_reu/bt1.jpg',
        queQuan: 'Nha Trang, Khánh Hòa',
        soThich: 'Xem phim, nghe nhạc, cafe với bạn bè, chụp ảnh trong giờ
rảnh rỗi'
    );
}

```

- Hiện thị ngày sinh từ đối tượng ngaySinh:

```

Text(
    '${myProfile.ngaySinh.day}/${myProfile.ngaySinh.month}/${myProfile.ngaySinh.year}',
    style: TextStyle(fontSize: 18,),
),

```

- Thiết kế trang 2 là một StatefulWidget có một thuộc tính là một đối tượng Profile:

```

class ProfileEditPage extends StatefulWidget {
    Profile profileEdit;
    ProfileEditPage(this.profileEdit);

    @override
    _ProfileEditPageState createState() => _ProfileEditPageState();
}

```

- Phương thức xử lý sự kiện khi bấm vào nút “Chỉnh sửa”:

```

Future<void> _editProfile() async{
    // Mở trang ProfileEditPage và chờ trả về một đối tượng Profile
    Profile editedProfile = await Navigator.push(
        context, MaterialPageRoute(
            builder:(context) => ProfileEditPage(myProfile),
        )
    );
    // Gọi phương thức setState để cập nhật giao diện ứng dụng
    setState(() {
        myProfile.hoTen = editedProfile.hoTen;
        myProfile.ngaySinh = editedProfile.ngaySinh;
        myProfile.queQuan = editedProfile.queQuan;
        myProfile.soThich = editedProfile.soThich;
    });
}

```

- Sử dụng TextField nhập văn bản:

```

TextField buildTextField(String label, TextEditingController controller,
bool readOnly) {
    return TextField(
        controller: controller,
        readOnly: readOnly ,
        maxLines: null, // để có thể nhập nhiều dòng trong TextField
        decoration: InputDecoration(

```

```

        labelText: label,
        border: OutlineInputBorder(
          borderRadius: BorderRadius.circular(5)
        ),
      ),
    );
  }
}

```

- Sử dụng phương thức `showDatePicker` để hiển thị lịch chọn ngày:

```

Future<DateTime> _selectDate(DateTime ngaySinh) async{
  final DateTime _pickedDate = await showDatePicker(
    context: context,
    initialDate: ngaySinh,
    firstDate: DateTime.now().subtract(Duration(days: 365*50)),
    lastDate: DateTime.now().add(Duration(days: 365*50)),
  );
  return _pickedDate;
}

```

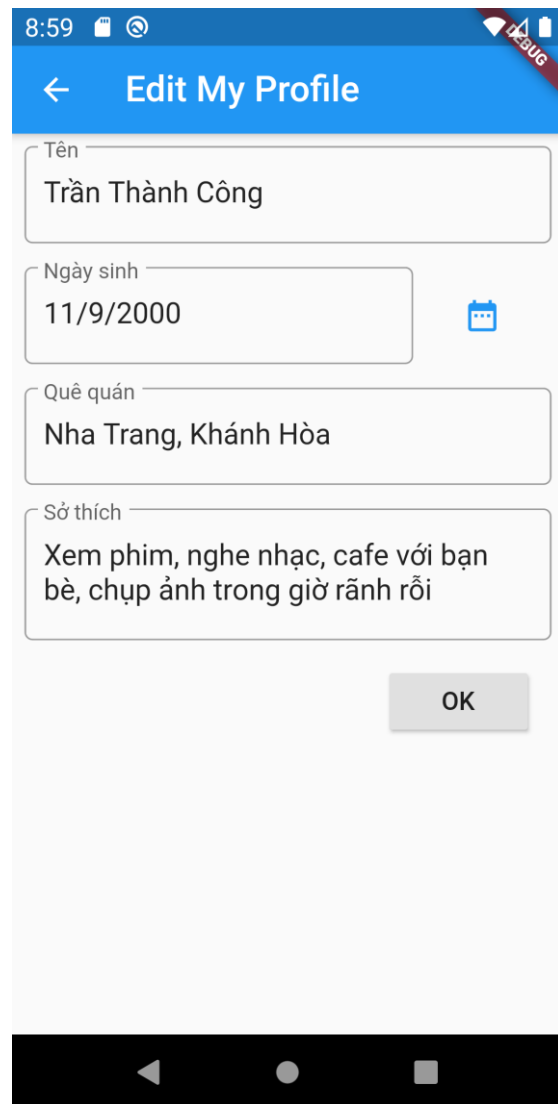
- Sử dụng Row Layout để hiển thị TextField và FlatButton trên cùng một dòng. Sử dụng Expanded để bọc TextField.

```

Row(
  children: [
    Expanded(child: buildTextField("Ngày sinh", ngaySinhController, true)),
    FlatButton(
      onPressed: () async{
        DateTime _selectedDate = await
          _selectDate(widget.profileEdit.ngaySinh);
        setState(() {
          widget.profileEdit.ngaySinh = _selectedDate;
          ngaySinhController.text = _textFromDate(_selectedDate);
        });
      },
      child: Icon(Icons.date_range, color: Colors.blue,)
    ),
  ],
),

```

-



Chú ý: Nên bọc tất cả các Widget của ứng dụng vào một SingleChildScrollView để giao diện vẫn hiển thị đúng và không bị lỗi khi nhập nội dung quá dài hoặc bàn phím của ứng dụng xuất hiện thêm khi nhập liệu.

Bài tập 5: Tương tự như bài tập 2, Sinh viên làm theo 3 version:

version 1: Dữ liệu lưu bằng SharedPreferences, quản lý trạng thái bằng setState.

version 2: Dữ liệu lưu bằng SharedPreferences, quản lý trạng thái bằng Provider.

version 3: Dữ liệu lưu bằng file .json

Hướng dẫn:

1. Version 1:

- Cài đặt lớp ProfilePreference hỗ trợ việc đọc, ghi dữ liệu preference.

- Các trang ứng dụng được cài đặt như sau:

```
class PreferenceSetStateApp extends StatelessWidget {
  @override
```

```

Widget build(BuildContext context) {
  return MaterialApp(
    home: ProfilePreferenceSetStatePage(),
  );
}

class ProfilePreferenceSetStatePage extends StatefulWidget {
  @override
  _ProfilePreferenceSetStatePageState createState() =>
  _ProfilePreferenceSetStatePageState();
}

class _ProfilePreferenceSetStatePageState extends
State<ProfilePreferenceSetStatePage> {
  Profile profile;
  bool loadData = false;
  bool error = false;
  @override
  Widget build(BuildContext context) {
    if(!error)
      if(loadData)
        return buildProfileWidget();
      else
        return buildLoadingWidget();
    else
      return
        errorBuildWidget();
  }

  void _loadData() async{
    try{
      profile = await ProfilePreference.readPreference();
      setState(() {
        loadData = true;
      });
    }catch(e){
      setState(() {
        error = true;
      });
      print("Lỗi đọc dữ liệu");
    }
  }

  Widget buildProfileWidget() {}

  Widget buildLoadingWidget() {}

  Widget errorBuildWidget() {}
}

```

2. Version 2:

Cài đặt lớp SharedPrefHelper kế thừa từ lớp ChangeNotifier:

```

class SharedPrefHelper extends ChangeNotifier{
  Profile _profile = Profile();

  SharedPrefHelper(){
    profile.hoTen = "Chưa có tên";
  }
}

```

```

        profile.queQuan = "Nhập quê quán";
        profile.ngaySinh = DateTime.now();
        profile.soThich = "Thích đủ thứ";
        profile.imageAssest = 'assets/bai_reu/bt1.jpg';
    }

    void getSharedProfile() async{
        SharedPreferences sharedPreferences = await SharedPreferences.getInstance();
        profile.hoTen = sharedPreferences.getString('hoTen') ?? "Chưa có tên";
        profile.queQuan = sharedPreferences.getString("queQuan") ?? "Nhập quê quán";
        profile.ngaySinh = sharedPreferences.getString("ngaySinh")!=null?
            DateTime.parse(sharedPreferences.getString("ngaySinh")) : DateTime.now();
        profile.soThich = sharedPreferences.get("soThich") ?? "Thích đủ thứ";
        notifyListeners();
    }

    Profile get profile => _profile;

    void updateProfile(Profile newProfile) async{
        _profile = newProfile;
        SharedPreferences sharedPreferences = await SharedPreferences.getInstance();
        sharedPreferences.setString("hoTen", _profile.hoTen);
        sharedPreferences.setString('queQuan', _profile.queQuan);
        sharedPreferences.setString('ngaySinh', _profile.ngaySinh.toString());
        sharedPreferences.setString('soThich', _profile.soThich);
        notifyListeners();
    }
}

```

Widget ứng dụng là một StatelessWidget có phương thức build trả về một ChangeNotifierProvider:

```

class ProfilePreferenceProviderApp extends StatelessWidget {
    @override
    Widget build(BuildContext context) {
        return ChangeNotifierProvider<SharedPrefHelper>(
            create:(context) {
                SharedPrefHelper sharedPrefHelper = SharedPrefHelper();
                sharedPrefHelper.getSharedProfile();
                return sharedPrefHelper;
            },
            builder:(context, child) => MaterialApp(
                title: "Profile version 2",
                theme: ThemeData(
                    primarySwatch: Colors.blue,
                    visualDensity: VisualDensity.adaptivePlatformDensity,
                ),
                home: ProfilePageVer2(),
            ),
        );
    }
}

```

Lớp ProfilePageVer2 chỉ cần là một StatelessWidget:

```

class ProfilePageVer2 extends StatelessWidget {
    @override

```

```

Widget build(BuildContext context) {
  double size = MediaQuery.of(context).size.width*0.75;
  SharedPreferences prefHelper = context.watch<SharedPreferences>();
  Profile myProfile = prefHelper.profile;
  .....
// Phương thức được gọi trong sự kiện nút bấm “Chỉnh sửa”
Future<void> _editProfile(BuildContext context, Profile profile) {
  // Mở trang ProfileEditPage và chờ trả về một đối tượng Profile
  Navigator.push(
    context, MaterialPageRoute(
      builder:(context) => ProfileEditPageVer2(profile),
    )
  );
}
}

```

- ProfileEditPageVer2 là một StatefulWidget có sự kiện nút bấm “Ok” như sau:

```

RaisedButton(
  onPressed: () {
    SharedPreferences sharePref = context.read<SharedPreferences>();
    _savedProfile(sharePref);
  },
  child: Text("OK"),
),

void _savedProfile(SharedPreferences sharedPrefHelper) {
  widget.profileEdit.ten = tenController.text;
  widget.profileEdit.queQuan = queQuanController.text;
  widget.profileEdit.soThich = soThichController.text;
  sharedPrefHelper.updateProfile(widget.profileEdit);
  Navigator.pop(context);
}

```

3. Version 3:

Cài đặt lớp hỗ trợ đọc ghi file: Sử dụng Template Method để cài đặt.

```

import 'dart:io';
import 'package:path_provider/path_provider.dart';
// Lớp được cài đặt chung cho tất cả các trường hợp đọc/ghi file text.
abstract class FileHelper{
  String fileName;

  FileHelper({this.fileName});

  String initialContent(); // Phương thức trừu tượng sẽ được implement ở các lớp con

  Future<String> get _localPath async{
    var directory = await getApplicationDocumentsDirectory();
    return directory.path;
  }

  Future<File> get _localFile async{
    String path = await _localPath;
    return File('$path/$fileName');
  }

  Future<File> writeString(String str) async{
    File file = await _localFile;

```

```

        return file.writeAsString(str);
    }

Future<String> readFile() async{
    try{
        File file = await _localFile;
        if(!file.existsSync()) {
            print('file chưa tồn tại: ${file.absolute}');
            await file.writeAsString(initialContent());
        }
        String content = await file.readAsString();
        return content;
    } catch(e){
        print('Lỗi khi đọc file');
        return null;
    }
}
}

```

// Lớp cài đặt riêng cho từng trường hợp.

```

const String congViecFile = "profile.json";
class FileProfileHelper extends FileHelper{

    FileProfileHelper():super(fileName: congViecFile);
    @override
    String initialContent() {
        return '{"profile":{}}';
    }
}

```

- Các lớp quản lý dữ liệu:

```

class ProfileJson{
    String hoTen;
    String queQuan;
    DateTime ngaySinh;
    String soThich;
    String imageAsset;

    ProfileJson(
        {this.hoTen, this.queQuan, this.ngaySinh, this.soThich, this.imageAsset});

    factory ProfileJson.fromJson(Map<String, dynamic> json){
        return ProfileJson(
            hoTen: json['hoTen'],
            queQuan: json['queQuan'],
            ngaySinh: DateTime.parse(json['ngaySinh']),
            soThich: json['soThich'],
            imageAsset: json['imageAsset']
        );
    }

    Map<String, dynamic> toJson(){
        return {
            'hoTen':this.hoTen,
            'queQuan':this.queQuan,
            'ngaySinh': ngaySinh,
            'soThich':this.soThich,
            'imageAsset':this.imageAsset
        };
    }
}

```

```

}

class ProfileDatabase extends ChangeNotifier{
  ProfileJson _profile;
  FileProfileHelper fileHelper = FileProfileHelper();
  ProfileDatabase(){
    _profile = ProfileJson(
      hoTen: "",
      ngaySinh: DateTime.now(),
      queQuan: '',
      soThich: '',
      imageAsset: "assets/bai_reu/bt1.jpg"
    );
  }

  ProfileJson get profile => _profile;

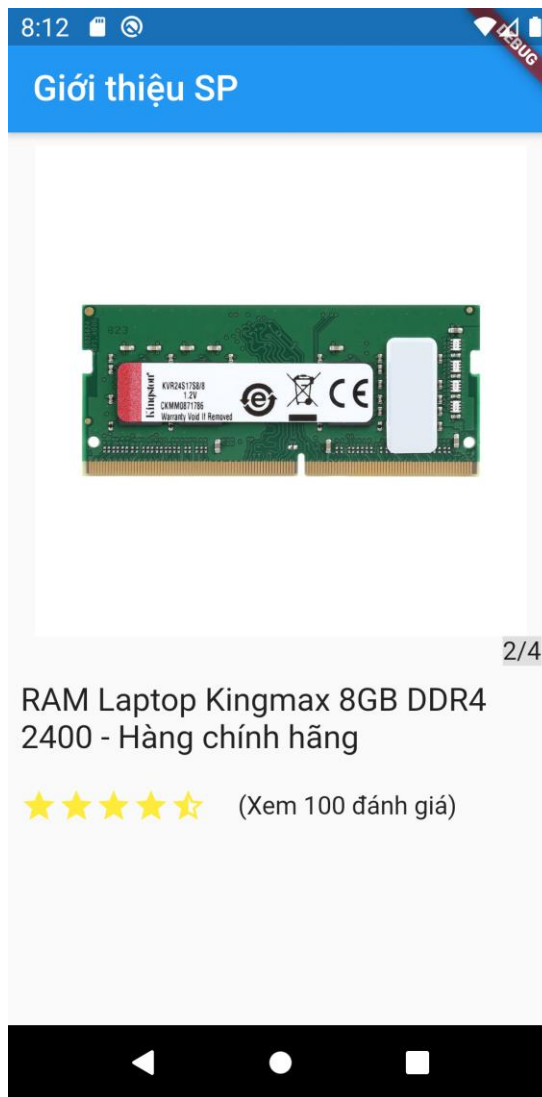
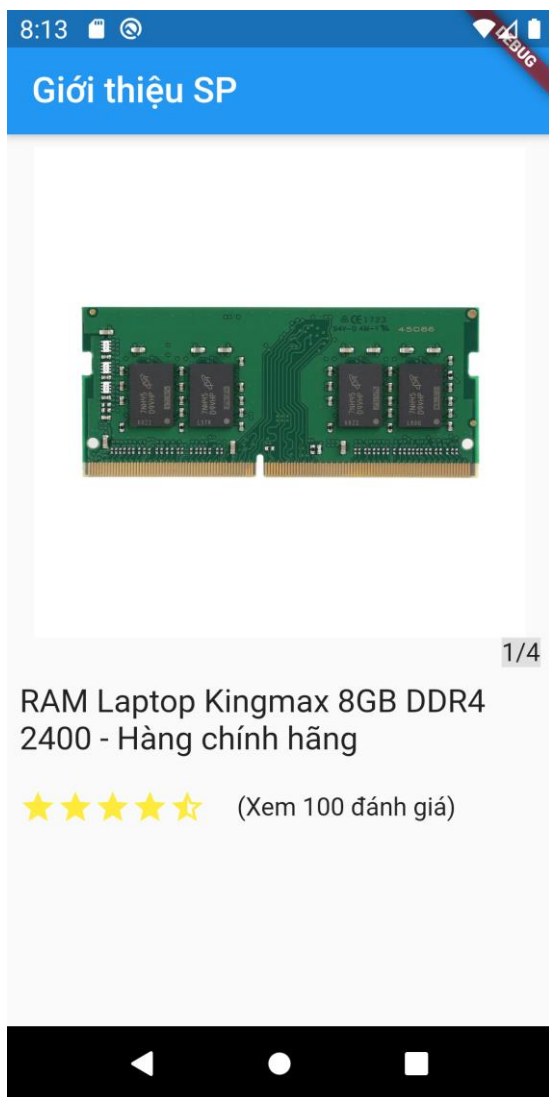
  void readProfile() async{
    String content = await fileHelper.readFile();
    Map data = json.decode(content)['profile'];
    if(data.isNotEmpty) {
      _profile = ProfileJson.fromJson(data);
      notifyListeners();
    }
  }

  void updateProfile(ProfileJson newProfile) async{
    _profile = newProfile;
    notifyListeners();
    String content = json.encode(newProfile);
    await fileHelper.writeString(content);
  }
}

```

- Tương tự như Version 2, sử dụng ChangeNotifierProvider để bọc trang ứng dụng (MaterialApp widget)

Bài tập 6: Sử dụng GestureDetector Thiết kế giao diện giới thiệu sản phẩm của một trang TMDT



Hướng dẫn:

- Sử dụng CarouselSlider (pub.dev: carousel_slider) để hiển thị các ảnh. Các ảnh có thể xuất hiện theo hiệu ứng trượt (slide) khi người sử dụng thực hiện thao tác vuốt trên màn hình. Khai báo package này trong tập tin pubspec.yaml
- Sử dụng StatefulWidget để thiết kế trang ứng dụng (Sinh viên tự giải thích vì sao không sử dụng StatelessWidget để thiết kế).
- Copy các ảnh của ứng dụng trong assets, khai báo các đường dẫn trong tập tin pubspec.yaml.
- Các state trong lớp State của ứng dụng:

```
class _SanPhamPageState extends State<SanPhamPage> {
  List<String> images = [
    'assets/ram/ram1.jpg',
    'assets/ram/ram2.jpg',
    'assets/ram/ram3.jpg',
    'assets/ram/ram4.jpg',
  ];
  int imagePos=0;
}
```

Hiển thị ảnh bằng CarouselSlider:

```

CarouselSlider.builder(
  itemCount: images.length,
  options: CarouselOptions(
    height: with_image*0.9,
    viewportFraction: 1,
    onPageChanged: (index, reason) {
      // Cập nhật số thứ tự ảnh
      setState(() {
        imagePos = index;
      });
    },
  ),
  itemBuilder: (context, index) {
    return Container(
      height: with_image*0.9, width: with_image*0.9,
      //margin: EdgeInsets.all(5),
      child: Image.asset(images[index]),
    );
  },
),

```

Hiển thị số thứ tự ảnh:

```

Row(
  children: [
    Expanded(child: SizedBox()),
    Container(
      color: Colors.grey[300],
      child: Text('${imagePos+1}/${images.length}'),
    ),
  ],
),

```

Bài tập 7: Form, Dialog

Hướng dẫn:

- Chỉ cần sử dụng StatelessWidget để thiết kế trang ứng dụng
- Sử dụng Form để bọc các TextFormField, DropdownButtonFormField.
- Khai báo một GlobalKey, sau đó gán cho thuộc tính key của Form để có thể truy xuất đến Form:

```
GlobalKey<FormState> _formState = GlobalKey<FormState>();
```

- Khai báo đối tượng MatHang để chứa thông tin nhập trên Form

```

class MatHang{
  String tenMH,loaiMH;
  int soLuong;
  MatHang({this.tenMH, this.loaiMH, this.soLuong});
}

List<String> loaiMHs = ['Tivi', 'Điện thoại', 'Laptop'];

class MyForm extends StatelessWidget {
  GlobalKey<FormState> _formState = GlobalKey<FormState>();
  MatHang mh = MatHang();
  ...
}

```



```
body: Form(
  key: _formState,
  autovalidateMode: AutovalidateMode.disabled,
  ...
)
```

- TextFormField mặt hàng được thiết kế như sau:

```
TextFormField(
  decoration: InputDecoration(
    labelText: 'Mặt hàng',
    border: OutlineInputBorder(
      borderRadius: BorderRadius.all(Radius.circular(10))
    )
  ),
  onSaved: (newValue) {mh.tenMH=newValue;},
  validator: (value) => value.isEmpty ? "Chưa có tên mặt hàng" : null,
),
```

Callback onSaved chỉ được gọi khi `_formState.currentState.save()` được gọi

- DropdownButtonFormField:

```
DropdownButtonFormField<String>(
  items: loaiMHs.map((loaiMH) => DropdownMenuItem<String>(
    child: Text(loaiMH),
    value: loaiMH,
  )).toList(),
  onChanged: (value) {
    mh.loaiMH=value;
  },
  validator: (value) => value ==null? "Chưa chọn loại mặt hàng" : null,
  decoration: InputDecoration(
    labelText: 'Loại mặt hàng',
    border: OutlineInputBorder(
      borderRadius: BorderRadius.all(Radius.circular(10))
    )
  ),
),
```

Phương thức *map* trên List chuyển một Item trong List thành một đối tượng cần thiết khác (trong trường hợp này là một `DropdownMenuItem<String>`)

- Thiết kế nút bấm và sự kiện trên nút bấm:

```
Row(
  children: [
    Expanded(child: SizedBox(), flex: 1,),
    RaisedButton(
      child: Text('OK'),
      color: Colors.white,
      onPressed: () {
        if(_formState.currentState.validate()) {
          _formState.currentState.save();
          _showAlertDialog(context);
        }
      },
    ),
    SizedBox(width: 20,),
  ],
)
```

```
void _showAlertDialog(BuildContext context){
  AlertDialog alertDialog = AlertDialog(
```

```

title: Text('Xác nhận'),
content: Text("Bạn nhập mặt hàng: ${mh.tenMH}\n"
  "Thuộc loại mặt hàng: ${mh.loaiMH}\n"
  "Với số lượng: ${mh.soLuong}"),
actions: [
  FlatButton(
    onPressed: () => Navigator.pop(context),
    child: Text('Ok'))
],
);
showDialog(
  context: context,
  builder:(context) => alertDialog,
);
}

```

4:57

Form Widget Demo

Mặt hàng

Chưa có tên mặt hàng

Loại mặt hàng

Điện thoại

Số lượng

3

OK

q w e r t y u i o p
a s d f g h j k l
↑ z x c v b n m ✕
?123 , 😊 . ✓

4:59

Form Widget Demo

Mặt hàng

Samsung Note 10

Loại mặt hàng

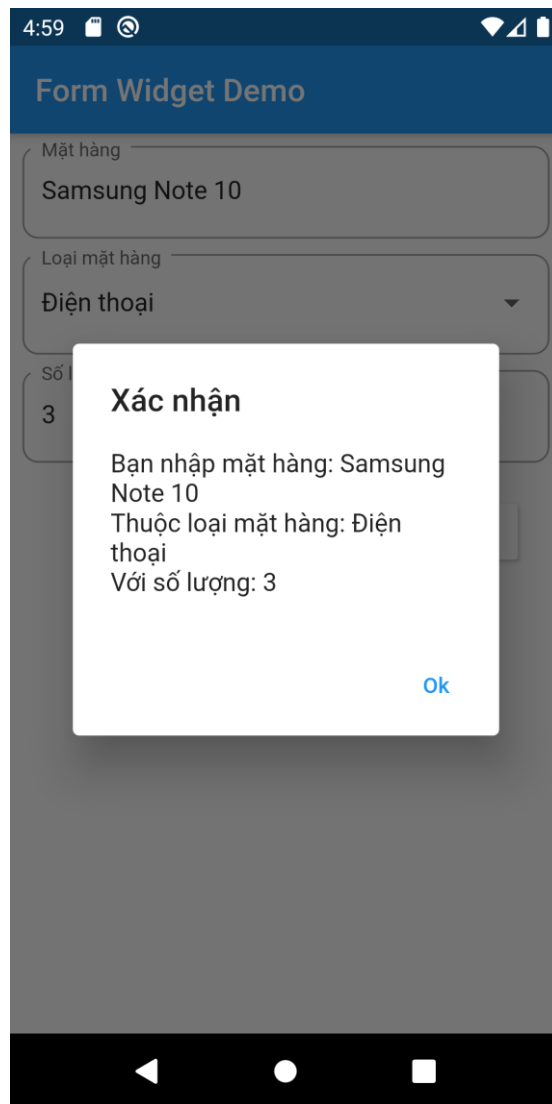
Điện thoại

Số lượng

3

OK

q w e r t y u i o p
a s d f g h j k l
↑ z x c v b n m ✕
?123 , 😊 . ✓



Bài tập 8: Xử lý dữ liệu Json và ListView

Đọc file json trong assets hoặc trong thư mục ứng dụng và hiển thị trên ListView.

Hướng dẫn:

- Tạo file users.json trong assets với nội dung như sau:

```
[
  {
    "name": "Tuấn",
    "email": "tuan@gmail.com"
  },
  {
    "name": "Thanh",
    "email": "thanh@gmail.com"
  },
  {
    "name": "Tùng",
    "email": "tung@gmail.com"
  },
  {
    "name": "Dũng",
    "email": "dung@gmail.com"
  },
]
```

```

    {
      "name": "Thảo",
      "email": "thao@gmail.com"
    }
  ]

```

- Để đọc file json thành một danh sách các đối tượng, ta sử dụng import hai thư viện:

```

import 'package:flutter/services.dart' show rootBundle;
import 'dart:convert';

```

Thư viện đầu tiên dùng để đọc file json, thư viện thứ hai dùng để encode và decode dữ liệu json.

- Khai báo hai lớp User như sau, phương thức khởi tạo factory User.fromJson tạo một User từ một đối tượng Map (dữ liệu key/value dạng json). Phương thức toJson hỗ trợ việc chuyển một đối tượng User thành một Map.

```

class User{
  String name, email;

  User({this.name, this.email});
  factory User.fromJson(Map<String, dynamic> json){
    return User(
      name: json['name'],
      email: json['email']
    );
  }

  Map<String, dynamic> toJson(){
    return{
      'name':name,
      'email':email
    };
  }
}

```

- Đọc file json và chuyển thành một danh sách các User:

```

Future<List<User>> fetchUserFromJson(String path) async{
  String userJson = await rootBundle.loadString(path);
  List<dynamic> list = jsonDecode(userJson) as List;
  return list.map((user) => User.fromJson(user)).toList();
}

```

- Sử dụng FutureBuilder để xây dựng giao diện ứng với dữ liệu được trả về bởi phương thức async

```

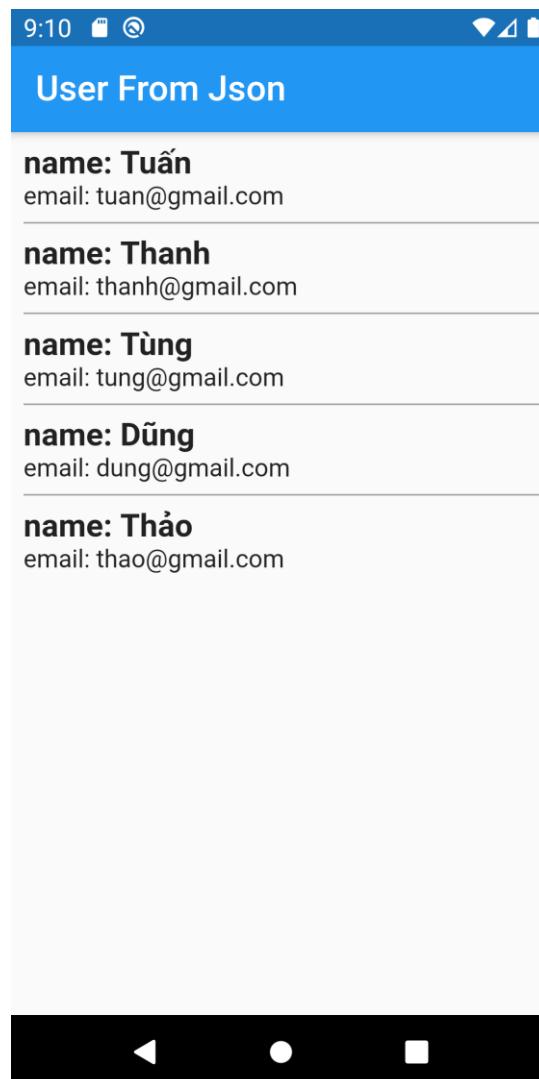
body: FutureBuilder<List<User>>(
  initData: [],
  future: fetchUserFromJson(path),
  builder: (context, snapshot) {
    if(snapshot.hasError)
      return Center(child:Text("Lỗi xảy ra"));
    else
      return snapshot.hasData?
        Padding(
          padding: const EdgeInsets.all(8.0),
          child: ListView.separated(
            itemBuilder:(context, index) => Column(
              crossAxisAlignment: CrossAxisAlignment.start,

```

```

        children: [
          Text('name: ${snapshot.data[index].name}',
            style: TextStyle(fontSize: 18, fontWeight: FontWeight.bold),
          ),
          Text('email: ${snapshot.data[index].email}'),
        ],
      ),
      separatorBuilder: (context, index) => Divider(
        color: Colors.grey,
        thickness: 1,
      ),
      itemCount: snapshot.data.length
    ),
  ),
): Center(child: CircularProgressIndicator());
},
),

```



Bài tập 8bis: Mở rộng bài tập 6: Cho phép chỉnh sửa, nhập thêm thông tin user.

Gợi ý:

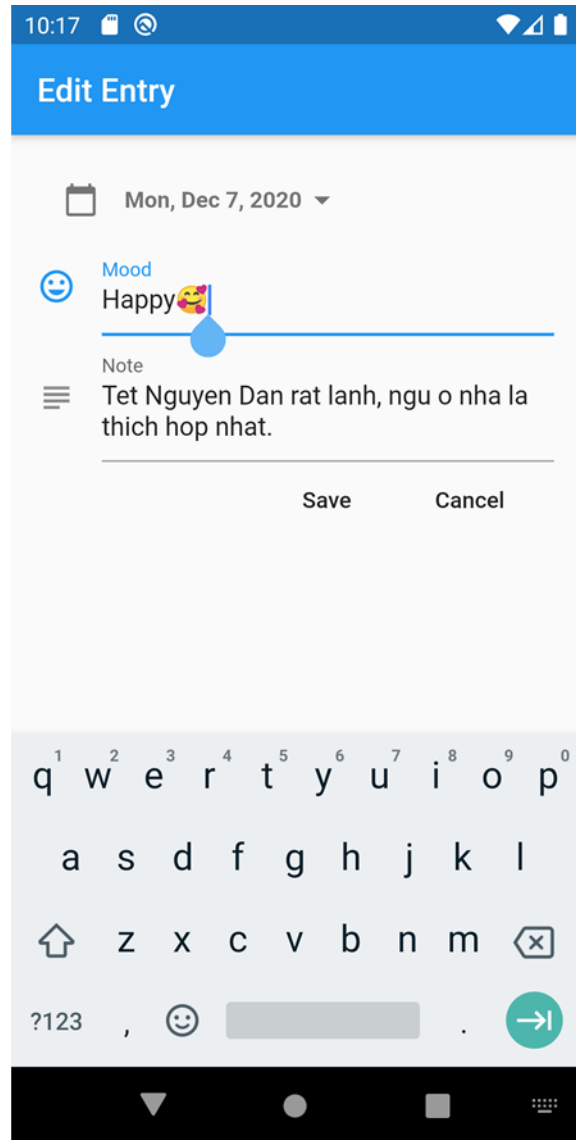
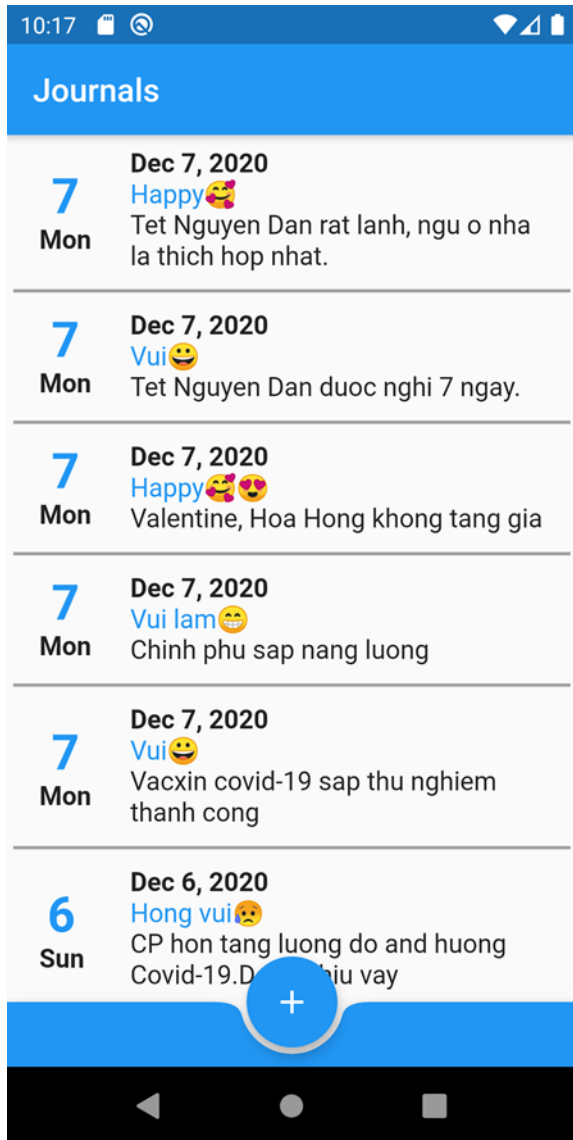
- Viết thêm lớp hỗ trợ đọc ghi file như bài tập số 4.

- Sử dụng Provider để quản lý trạng thái ứng dụng.

Bài tập 9: Đọc ghi file trong Flutter:

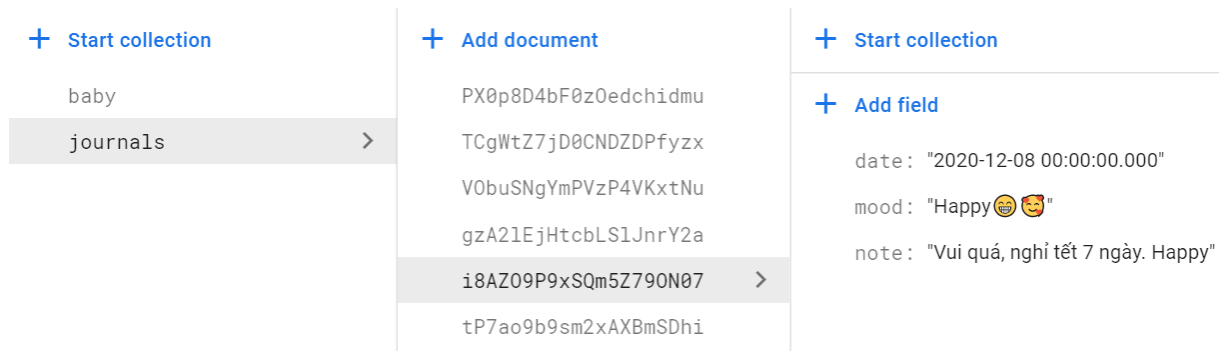
Thiết kế ứng dụng quản lý công việc trên Flutter, dữ liệu được tổ chức thành file json và ghi vào bộ nhớ cục bộ trên điện thoại.

Bài tập 10: Sử dụng Firebase Firestore:



Hướng dẫn:

1. Thiết lập kết nối firebase với ứng dụng.
2. Tạo một Collection có tên là "journals" và nhập vài document dữ liệu mẫu



3. Cài đặt các lớp dữ liệu:

3.1 Lớp JournalDoc có hỗ trợ jsonEncode và jsonDecode:

```
class JournalDoc{
  String date;
  String mood;
  String note;
  JournalDoc({this.date, this.mood, this.note});

  factory JournalDoc.fromJson(Map<String, dynamic> json)=>
    JournalDoc(
      date: json['date'],
      mood: json['mood'],
      note: json['note'],
    );

  Map<String, dynamic> toJson()=>
  {
    'date': date,
    'mood':mood,
    'note':note
  };
}
```

3.2 Lớp dữ liệu có chứa DocumentReference để cập nhật dữ liệu (Document):

```
class JournalSnapshot{
  JournalDoc doc;
  DocumentReference reference;
  JournalSnapshot({this.doc, this.reference});

  JournalSnapshot.fromJson(DocumentSnapshot snapshot):
    doc = JournalDoc.fromJson(snapshot.data()),
    reference = snapshot.reference;

  Future<void> update(String date, String mood, String note) async{
    return await reference.set({
      'date': date,
      'mood':mood,
      'note':note
    });
  }
}
```

3.3 Lớp tương tác với dữ liệu Firestore:

```
class FirestoreDatabaseJournal{
  Stream<List<JournalSnapshot>> getJournalsFromFirebase(){
    Stream<QuerySnapshot> streamQuerySnapshot =
```

```

FirebaseFirestore.instance.collection("journals").snapshots();
    return streamQuerySnapshot.map((QuerySnapshot querySnapshot) =>
        querySnapshot.docs.map((DocumentSnapshot documentSnapshot) =>
            JournalSnapshot.fromSnapshot(documentSnapshot)
        )).toList()
    );
}

Future<void> addJournal(JournalDoc journal) async{
    return await FirebaseFirestore.instance.collection("journals")
        .add({
            'date': journal.date,
            'mood': journal.mood,
            'note': journal.note
        }).then((value) => print("Đã thêm"));
}

Future<void> deleteJournal(JournalSnapshot journalSnapshot)
{
    return journalSnapshot.reference.delete();
}
}

```

4. Trang ứng dụng, khởi tạo firebase

```

class JournalAppPageFirebase extends StatefulWidget {
    @override
    _JournalAppPageFirebaseState createState() =>
    _JournalAppPageFirebaseState();
}

class _JournalAppPageFirebaseState extends State<JournalAppPageFirebase> {
    bool _initialized = false;
    bool _error = false;
    @override
    Widget build(BuildContext context) {
        if(_error)
            return Container(
                color: Colors.white,
                child: Center(
                    child: Text(
                        "Lỗi kết nối",
                        style: TextStyle(fontSize: 16),
                        textDirection: TextDirection.ltr,
                    ),
                ),
            );
        else
            if(_initialized)
                return Provider<FirestoreDatabaseJournal>(
                    create:(context) => FirestoreDatabaseJournal(),
                    child: MaterialApp(
                        debugShowCheckedModeBanner: false,
                        title: 'Journal App Firebase',
                        theme: ThemeData(
                            primarySwatch: Colors.blue,
                            bottomAppBarColor: Colors.blue,
                        ),
                        home: JournalPageProvider(),
                    ),
                );
            else
                return Container(
                    color: Colors.white,
                    child: Center(
                        child: Text(
                            "Chưa khởi tạo",
                            style: TextStyle(fontSize: 16),
                            textDirection: TextDirection.ltr,
                        ),
                    ),
                );
    }
}

```



```

        else return Container(
          color: Colors.white,
          child: Center(
            child: Text(
              "Đang kết nối",
              style: TextStyle(fontSize: 16),
              textDirection: TextDirection.ltr,
            ),
          ),
        );
      }

void initializeFlutterFire() async {
  try {
    await Firebase.initializeApp();
    setState(() {
      _initialized = true;
    });
  } catch (e) {
    _error = true;
  }
}

@override
void initState() {
  initializeFlutterFire();
  super.initState();
}
}

```

5. Để sử dụng `Stream<List<JournalSnapshot>>`, `JournalPageProvider` (là một `StatelessWidget`) sử dụng một `StreamBuilder<List<JournalSnapshot>>`:

```

Widget build(BuildContext context) {
  FirestoreDatabaseJournal fireStoreDatabaseJournal =
    Provider.of<FirestoreDatabaseJournal>(context);

  .....

  body: StreamBuilder<List<JournalSnapshot>>(
    stream: fireStoreDatabaseJournal.getJournalsFromFirebase(),
    builder: (context, snapshot) {
      if(snapshot.hasData)
        return _buildListViewSeparated(snapshot.data, context);
      return Center(child: Text("Đang tải dữ liệu"));
    },
  ),
)

```

Bài tập 11: SQLite

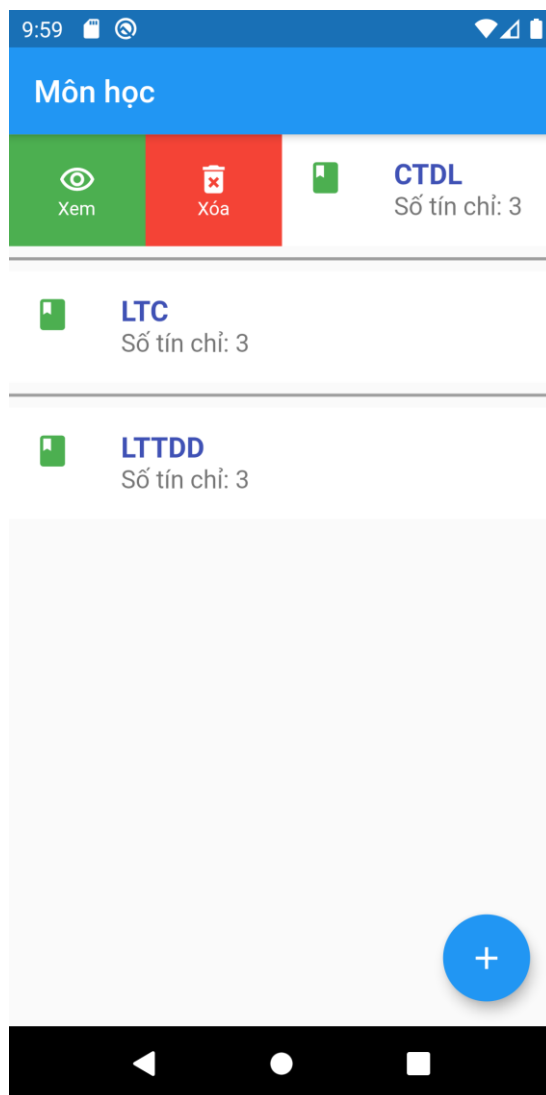
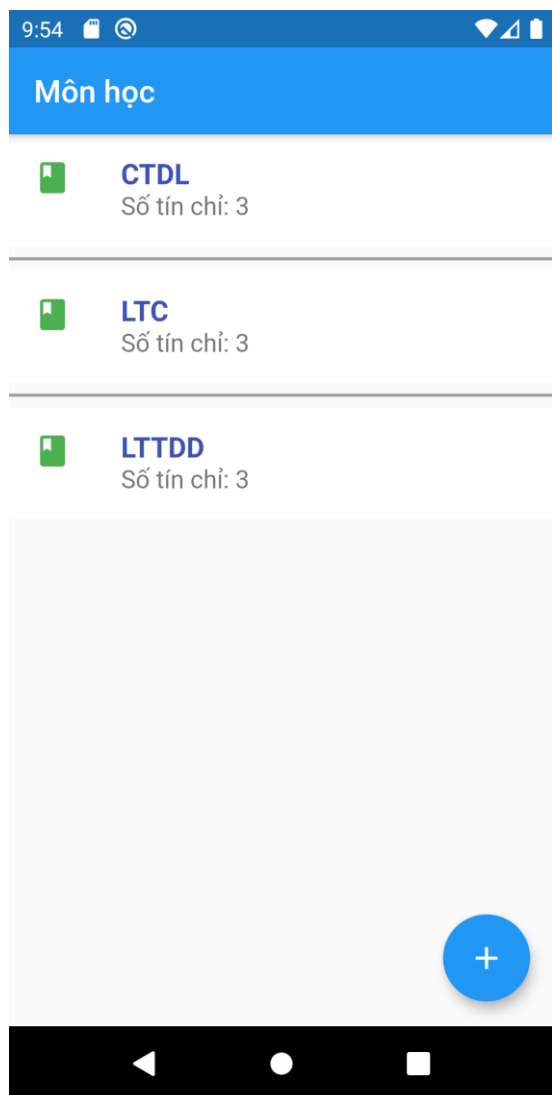
- Ứng dụng quản lý điểm các môn học

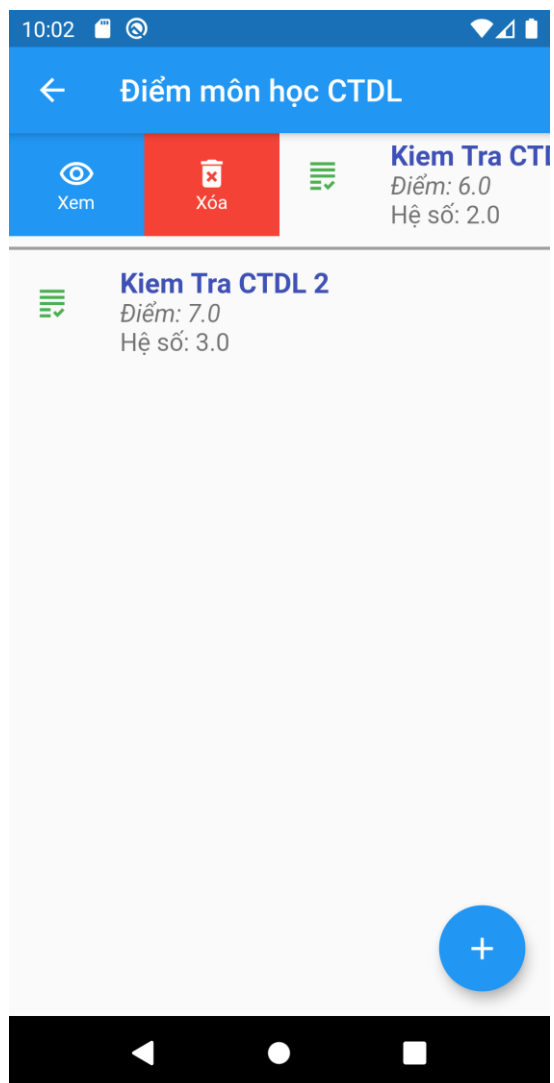
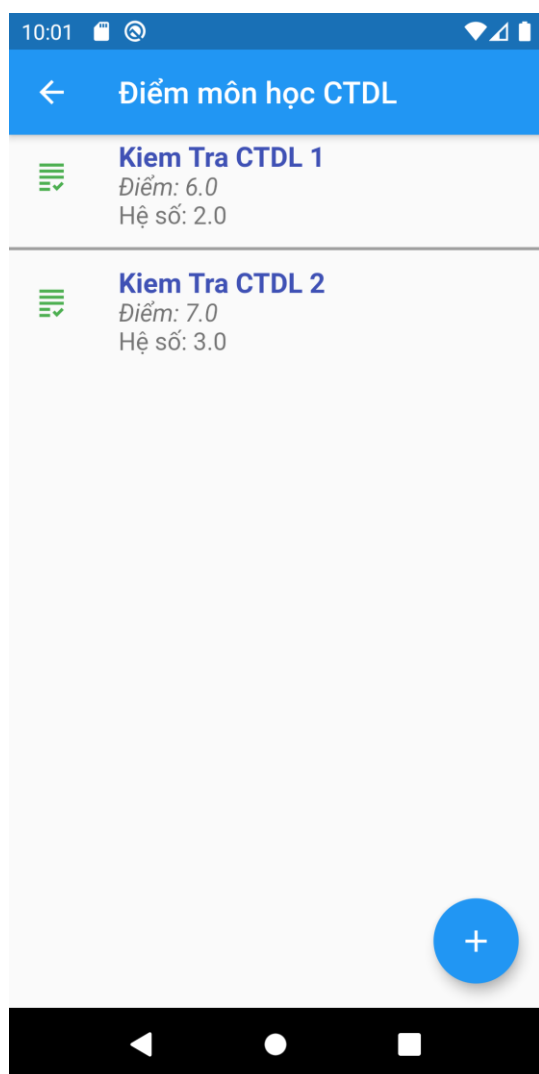
Hướng dẫn: Cài đặt các thư viện sau:

```

sqflite: ^1.3.2+2
path: ^1.7.0
provider: ^4.3.3
flutter_slidable: ^0.5.7

```





10:04 10:06

Cập nhật điểm **Nhập điểm**

Tên bài kiểm tra
Kiem Tra CTDL 1

Tên bài kiểm tra

Hệ số
2.0

Hệ số

Điểm
6.0

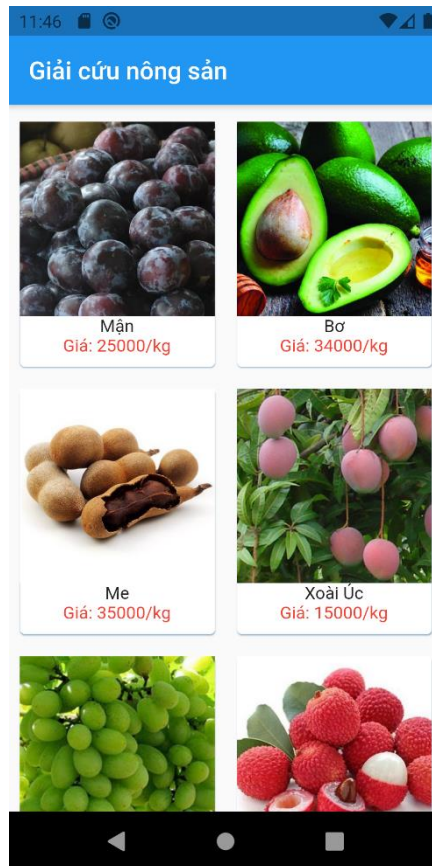
Điểm

Cập nhật Cancel Thêm Cancel

1 2 3 -
4 5 6 _
7 8 9 ×
, 0 . ✓

Bài tập 12:

Sử dụng GridView để hiển thị giao diện sau:



Hướng dẫn:

- Viết lớp Fruit gồm các thuộc tính sau để mô tả mỗi loại trái cây:
 - late String ten: Tên của trái cây
 - late Double gia: Giá của loại trái cây
 - late String url: Đường dẫn ảnh của loại trái cây
- List<Fruit> listSP: Khởi tạo danh sách của các loại trái cây.
- Hiển thị danh sách trái cây trên Gridview: Sử dụng GridView.extend

```

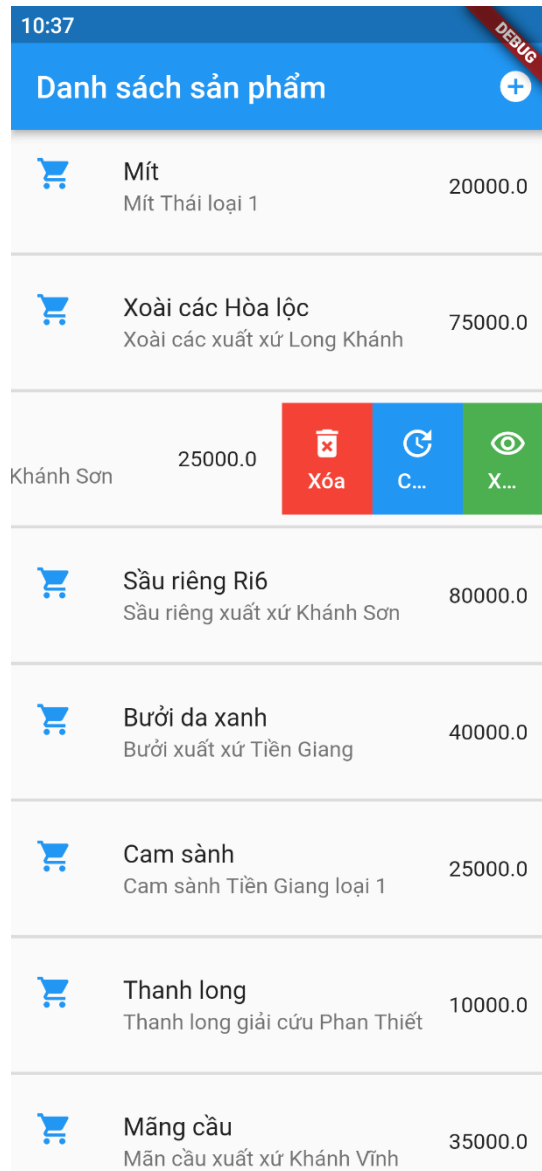
GridView.extend(
  padding: EdgeInsets.only(top: 10, bottom: 10, left: 5, right: 5),
  maxCrossAxisExtent: 250,
  mainAxisSpacing: 10,
  crossAxisSpacing: 10,
  childAspectRatio: 0.8,
  children: listSP.map((sp) => Card(
    elevation: 1,
    shadowColor: Colors.blue,
    child: Column(
      children: [
        Image.network(sp.url),
        Text("${sp.ten}"),
        Text("Giá: ${sp.gia}/kg",
          style: TextStyle(color: Colors.red),),
      ],
    ),
  ),
)

```

```
)).toList()
),
```

Bài tập 13:

Thiết kế ứng dụng để hiển thị danh sách sản phẩm theo giao diện sau đây



Hướng dẫn:

- Danh sách sản phẩm được lưu trong bộ nhớ RAM
- Sử dụng Provider để quản lý trạng thái là danh sách các sản phẩm như đã hướng dẫn trên lớp
- Sử dụng ListView để hiển thị danh sách dữ liệu (bọc ListView trong Consumer)
- Sử dụng thư viện Flutter Slidable để hiển thị từng mục của danh sách (Chú ý thuộc tính child của Slidable là một ListTile)

- Sinh viên có thể viết chức năng cho các nút hoặc không.

Bài tập tổng hợp 14:

Viết ứng dụng đọc báo RSS của trang VN Express.net



Hướng dẫn, sử dụng các thư viện sau đây:

- http: Dùng để đọc và lấy nội dung của một trang web từ một url.
- xml2json: Chuyển đổi nội dung RSS từ dạng XML sang JSON.
- webview_flutter: Hiển thị nội dung trang web từ một địa chỉ URL.
- provider: Quản lý trạng thái của ứng dụng: Nội dung của RSS.
- Sử dụng ListView để hiển thị nội dung của RSS
- Sử dụng WebView

- Đọc nội dung RSS và chuyển thành mảng các JSON Object:

```

abstract class RSSItem {
    String title;
    String pubDate;
    String description;
    String link;
    String imageUrl;
    RSSItem({this.title, this.pubDate, this.description, this.link, this.imageUrl});

    Map<String, dynamic> toJson(){
        return{
            "title":this.title,
            "date":this.pubDate,
            "description":this.description,
            "url":this.link,
            "imageUrl":this.imageUrl
        };
    }
}

class VNExpressRSSItem extends RSSItem{

    VNExpressRSSItem({String title, String pubDate, String description, String link,
String imageUrl})
        : super(title: title, pubDate: pubDate, description: description, link: link,
imageUrl: imageUrl);

    factory VNExpressRSSItem.fromJson(Map<String, dynamic> json){
        return VNExpressRSSItem(
            title: json['title'],
            pubDate: json['pubDate'],
            description: _getDescription(json['description']),
            link: json['link'],
            imageUrl: _getImageUrl(json['description']),
        );
    }

    static String _getImageUrl(String rawDescription){
        // 9: độ dài của pattern: img src="
        int start = rawDescription.indexOf('img src="') + 9;
        if(start>9)
        {
            int end = rawDescription.indexOf('"', start);
            return rawDescription.substring(start, end);
        }
        return null;
    }

    static String _getDescription(String rawDescription) {
        // 9: Độ dài của pattern </a></br>
        int start = rawDescription.indexOf('</a></br>') +9;
        if(start>9)
        {
            //int end = rawDescription.length -1;
            return rawDescription.substring(start);
        }
        return "";
    }
}

```


- Lớp Provider dùng để quản lý trạng thái của ứng dụng

```
class RSSProvider extends ChangeNotifier{
  String _rssURL ="https://vnexpress.net/rss/tin-moi-nhat.rss";
  List<RSSItem> _listRSSItem =[];

  Future<void> readRSSItems() async{
    var rssJsons = await _fetchRSS();
    if(rssJsons!=null) {
      //var List = json.decode(rssJson) as List;
      _listRSSItem= rssJsons.map((item)=>VNExpressRSSItem.fromJson(item)).toList();
      notifyListeners();
    }
  }

  // Getter
  List<RSSItem> get listRSSItems => _listRSSItem;

  Future<List<dynamic>> _fetchRSS() async{
    final response = await http.get(_rssURL);
    if(response.statusCode == 200){
      final xml2Json = Xml2Json();
      // Nếu sử dụng xml2Json.parse(response.body)
      xml2Json.parse(utf8.decode(response.bodyBytes));
      var rssJson = xml2Json.toParker();
      Map<String, dynamic> jsonData = jsonDecode(rssJson);
      return (jsonData["rss"]["channel"]["item"]);
    }
    return null;
  }
}
```

- Trang Ứng dụng RSS

```
class RSSAppPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return ChangeNotifierProvider(
      create: (context) {
        RSSProvider provider = RSSProvider();
        provider.readRSSItems();
        return provider;
      },
      child: MaterialApp(
        debugShowCheckedModeBanner: false,
        theme: ThemeData(
          primarySwatch: Colors.blue,
          visualDensity: VisualDensity.adaptivePlatformDensity,
        ),
        home: RSSPage(),
      ),
    );
  }
}
```

- Trang hiển thị RSS

```
class RSSPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    List<RSSItem> rssItems = context.select<RSSProvider, List<RSSItem>>((value) =>
```



```

    ),
  ),
);
}

Widget _getImage(String url){
  if(url!=null)
    return Image.network(url, fit: BoxFit.fitWidth,);
  return Center(
    child: Icon(Icons.image),
  );
}
}

```

- **Hiển thị trang web:**

```

class MyWebPage extends StatefulWidget {
  String url;
  MyWebPage({this.url});
  @override
  _MyWebPageState createState() => _MyWebPageState();
}

class _MyWebPageState extends State<MyWebPage> {
  final Completer<WebViewController> _controller = Completer<WebViewController>();
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("VNExpress.net"),
      ),
      body: WebView(
        initialUrl: widget.url,
        onWebViewCreated: (webViewController) {
          _controller.complete(webViewController);
        },
        onPageStarted: (String url) {
          print('Page started loading: $url');
        },
        onPageFinished: (String url) {
          print('Page finished loading: $url');
        },
        gestureNavigationEnabled: true,
      ),
    );
  }

  @override
  void initState() {
    super.initState();
    // Enable hybrid composition.
    if (Platform.isAndroid) WebView.platform = SurfaceAndroidWebView();
  }
}

```

Yêu cầu:

1. Vẽ Widget Tree của ứng dụng.
2. Cài đặt ứng dụng.

Bài tập tổng hợp 15:

Viết ứng dụng quản lý chi tiêu đơn giản trên Flutter

Yêu cầu: Dữ liệu quản lý local hoặc trên cloud.