Bài tập 1: 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.
- 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.

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: 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
rãnh rỗi'
```

```
    );
  }
```

- Hiển 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:

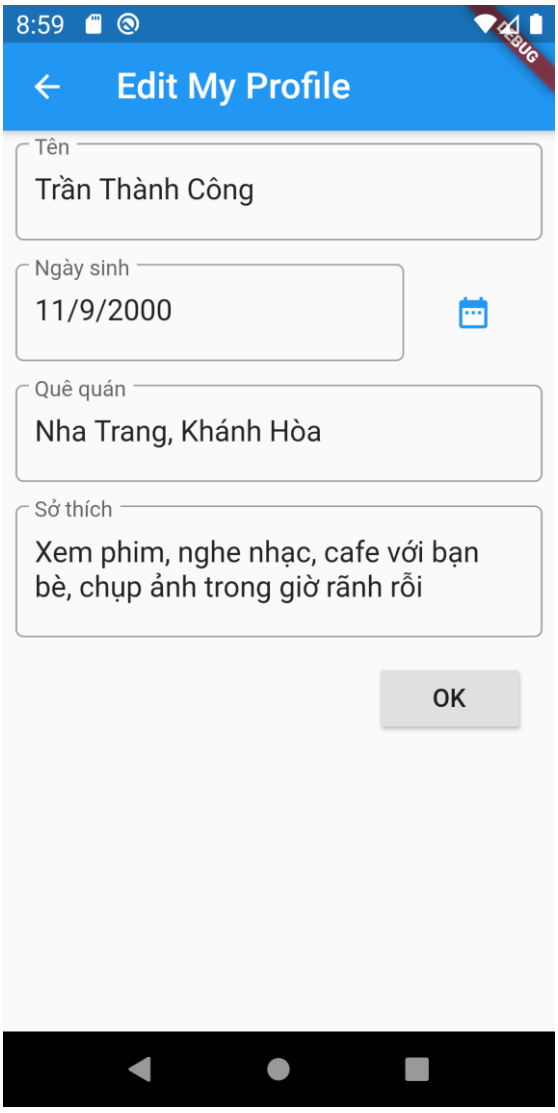
```
Future<DateTime> _selectDate(DateTime ngaySinh) async{
  final DateTime _pickedDate = await showDatePicker(
    context: context,
    initialDate:ngaySinh,
    firstDate: DateTime.now().subtract(Duration(days: 365*50)),
    lastDate: DateTime.now().add(Duration(days: 365*50)),
```

```
    );
    return _pickedDate;
  }
```

- Sử dụng Row Layout để hiện 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 3: 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 buildLoaddingWidget();
    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 buildLoaddingWidget() {}

        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;
    SharedPrefHelper prefHelper = context.watch<SharedPrefHelper>();
    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: () {
          SharedPrefHelper sharePref = context.read<SharedPrefHelper>();
          _savedProfile(sharePref);
        },
        child: Text("OK"),
      ),

      void _savedProfile(SharedPrefHelper sharedPrefHelper) {
        widget.profileEdit.hoTen = 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;
    }
  }
}
```

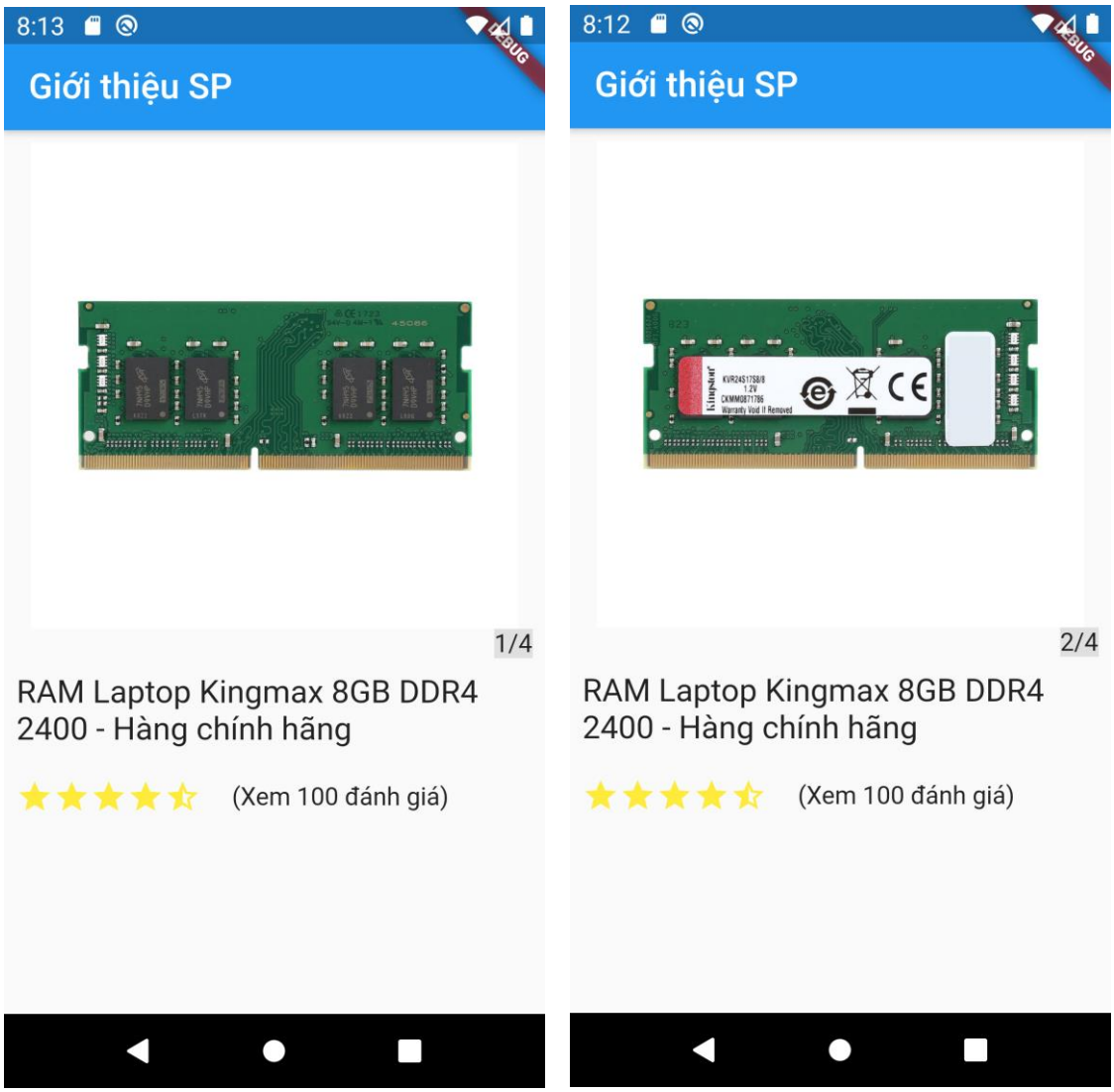
// Lớp cài đặt riêng cho từng trường hợp.

```
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 4: Sử dụng GestureDetector Thiết kế giao diện giới thiệu sản phẩm của một trang TMĐT



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 5: Form, Dialog

Hướng dẫn:

- Chỉ cần sử dụng StatelessWidget để thiết kế trang ứng dụng
- Sử dụng Form để bọc các TextFromField, 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))
      )
  ),),
```

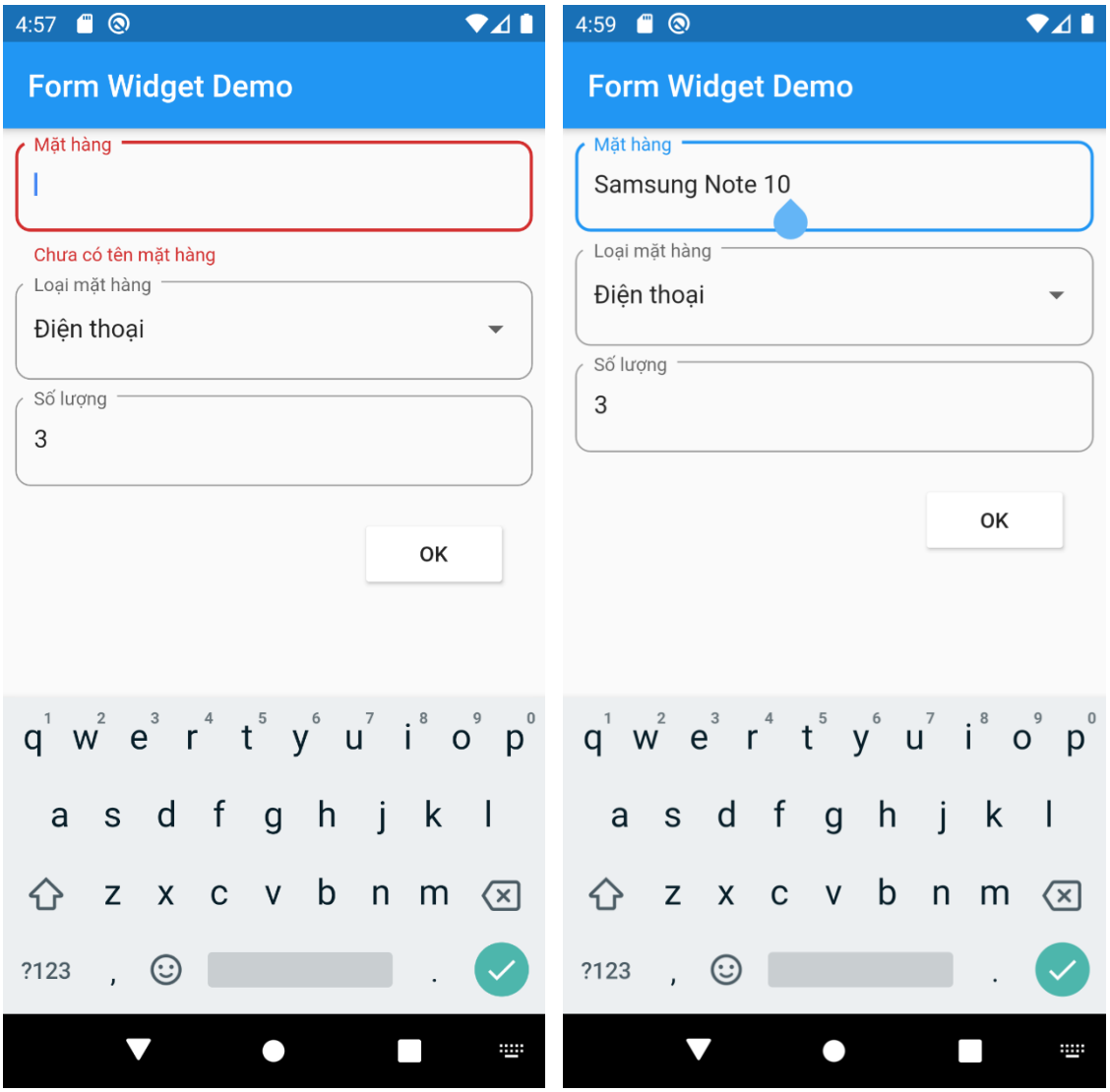
Phương thức *map* trên List chuyển một Item trong List thành một đối tượng cần thiết khác (trong trường hợp này là một DropdownMenuItem<String>)

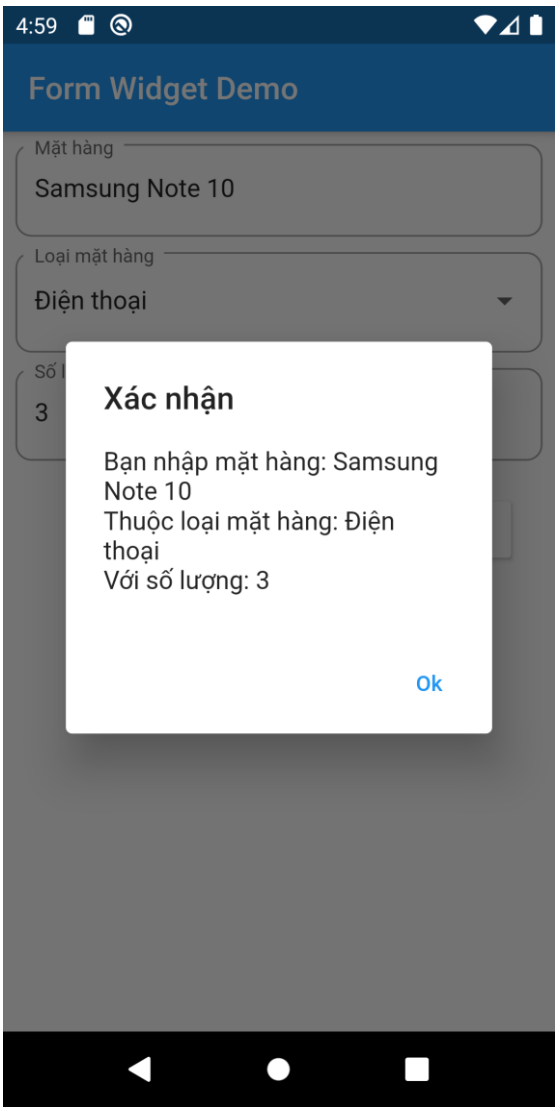
- Thiết kế nút bấm và sự kiện trên nút bấm:

```
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,
      );
    }
```

○





Bài tập 6: Xử lý dữ liệu Json và ListView

Đọc file json trong assets 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:

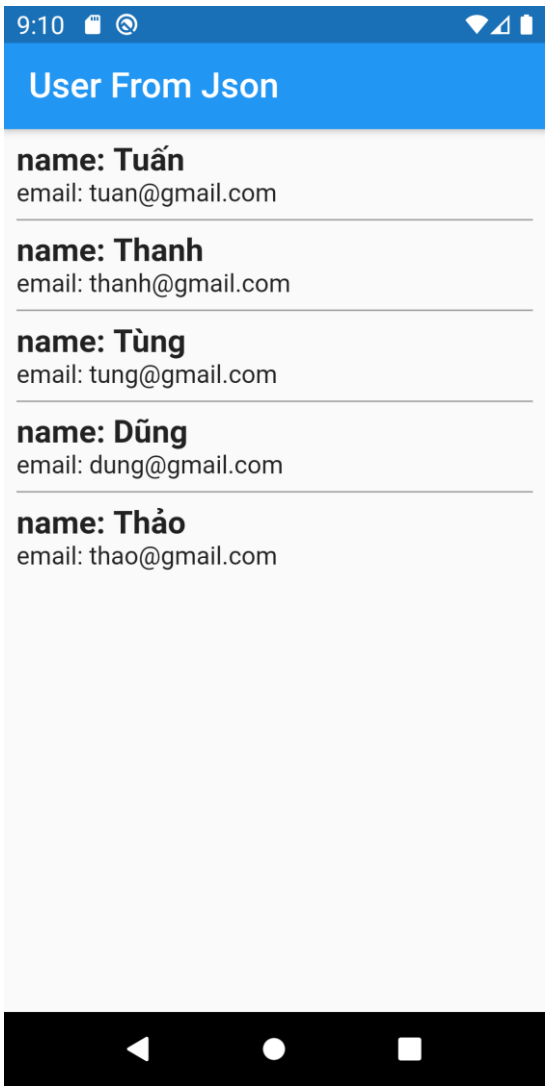
```
Future<List<User>> fetchUserFromJson(String path) async{
    String userJson = await rootBundle.loadString(path);
    List<dynamic>  list = jsonDecode(userJson) as List;
    return list.map((user) => User.fromJson(user)).toList();
}
```

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

```
body: FutureBuilder<List<User>>(
  initialData: [],
  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 6bis: Mở rộng bài tập 6: Cho phép chỉnh sửa, nhập thêm thông tin user.

Gợi ý:

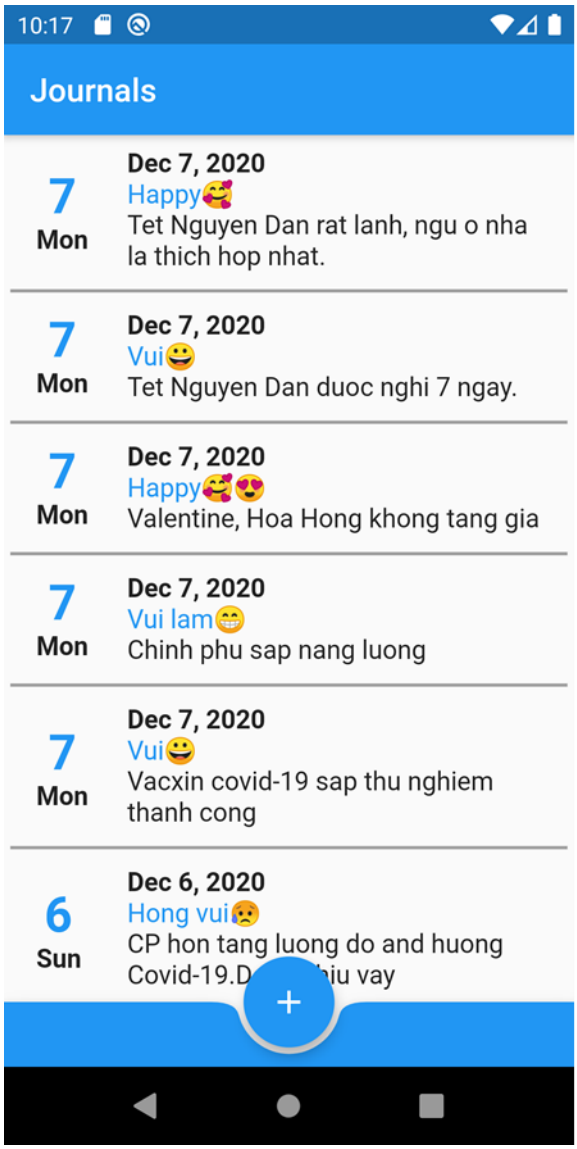
- Viết thêm lớp hỗ trợ đọc ghi file như bài tập số 3.

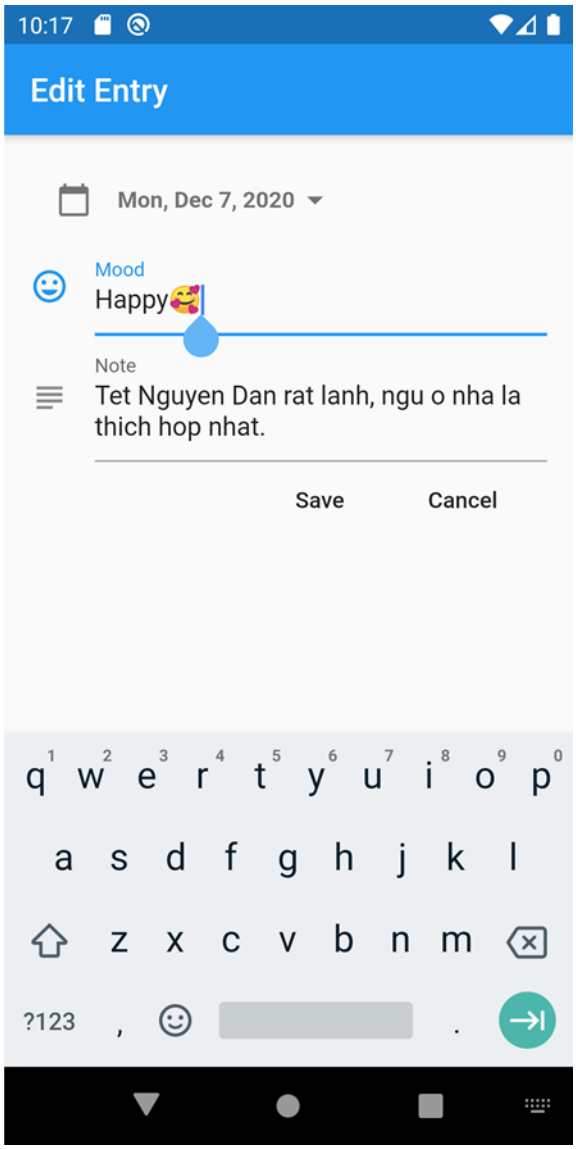
- Sử dụng Provider để quản lý trạng thái ứng dụng.

Bài tập 7: Đọc ghi file trong Flutter:

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

Bài tập 8: Sử dụng Firebase FireStore:

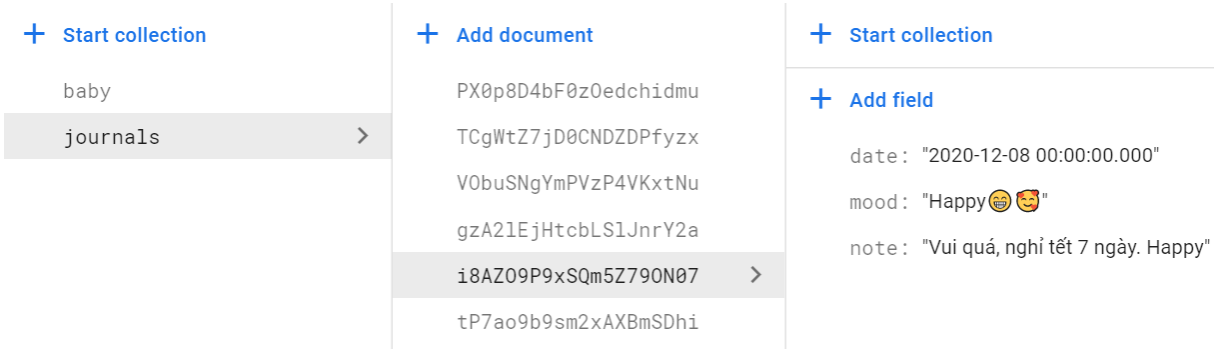




Hướng dẫn:

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.fromSnapshot(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(
                    "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>>:

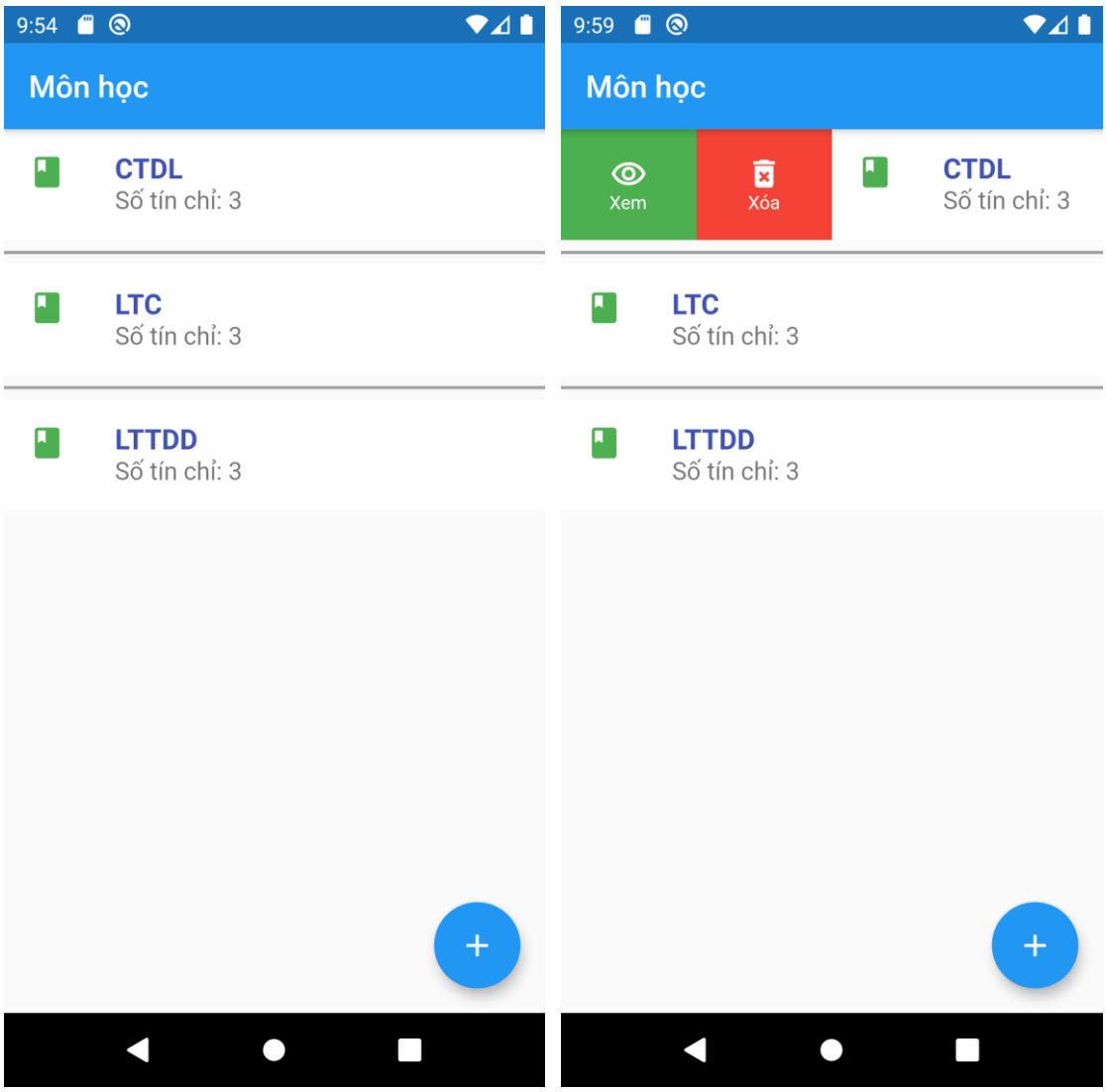
```
    Widget build(BuildContext context) {
      FireStoreDatabaseJournal fireStoreDatabaseJournal =
          Provider.of<FireStoreDatabaseJournal>(context);

    ……

    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"));
      },
    ),
```

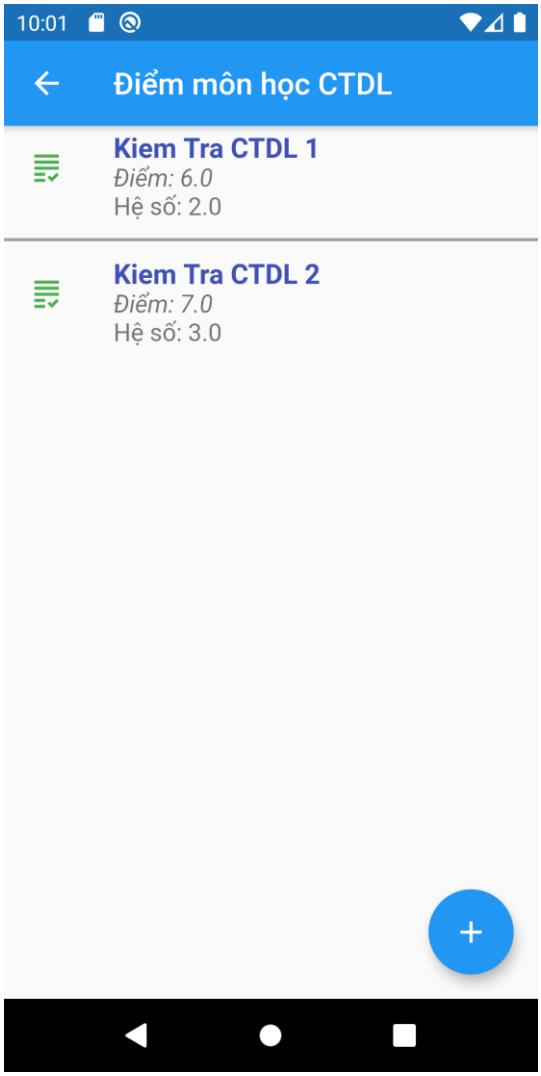
Bài tập 9: SQLite

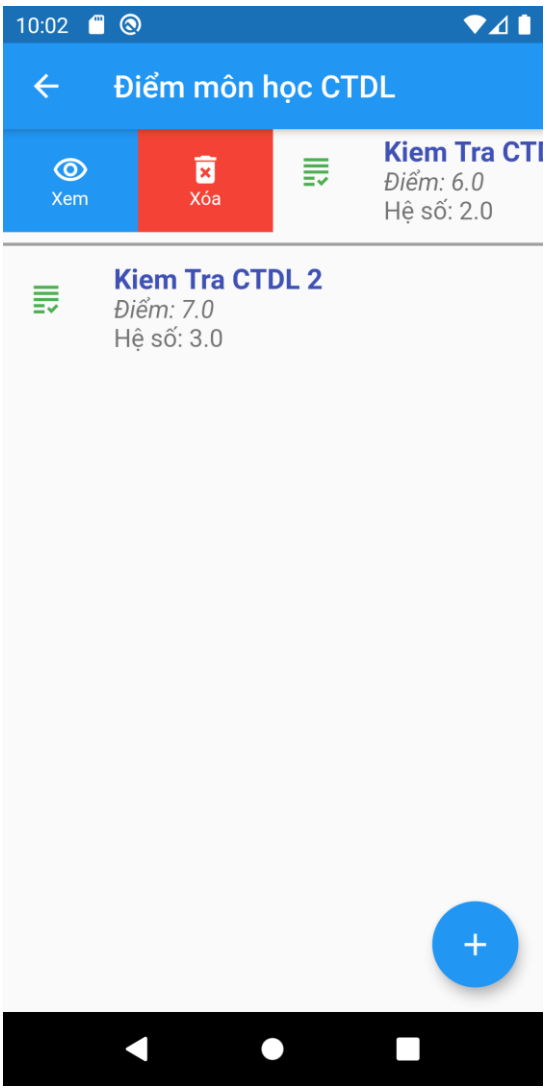
- Ứng dụng quản lý điểm các môn học

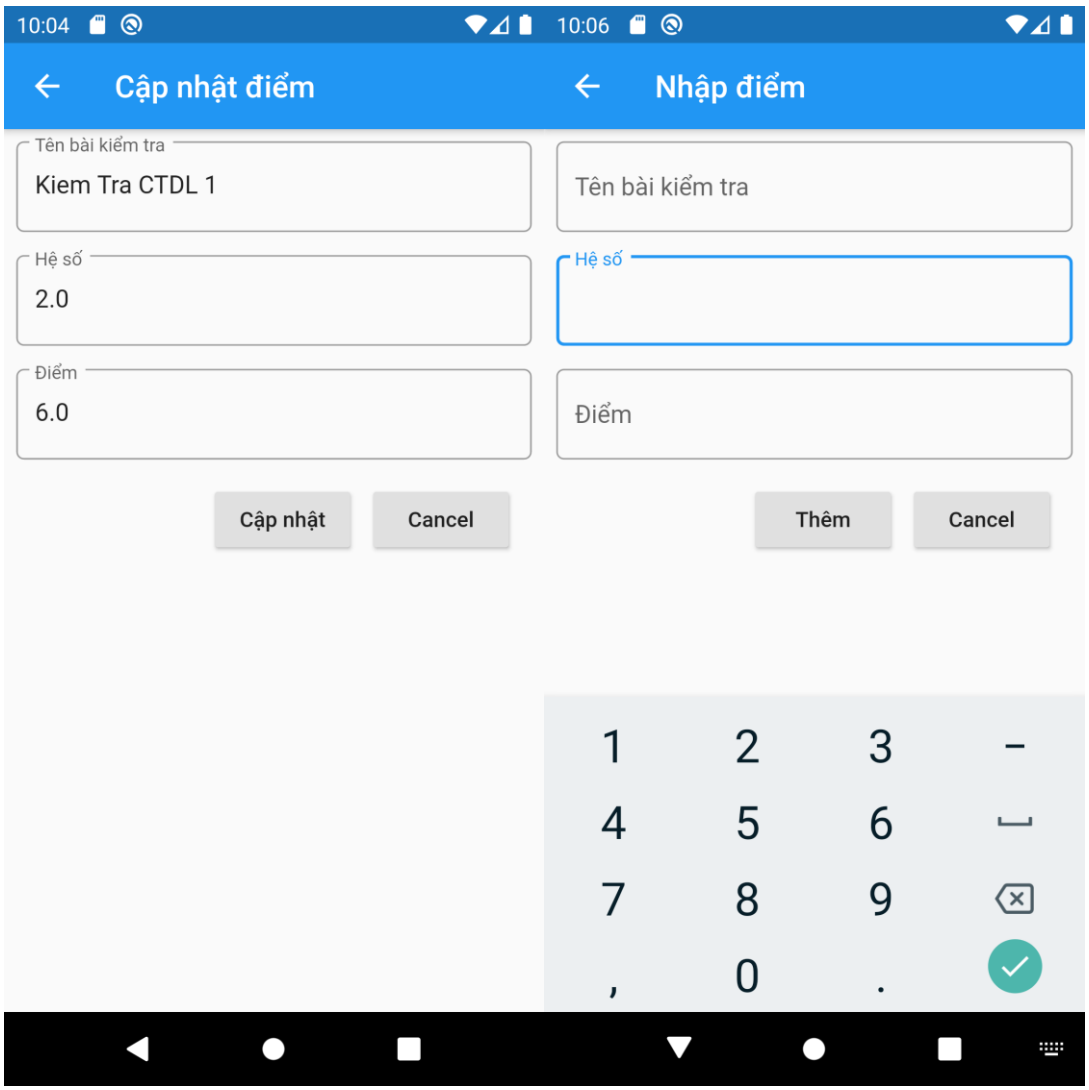
Hướng dẫn: Cài đặt các thư viện sau:

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

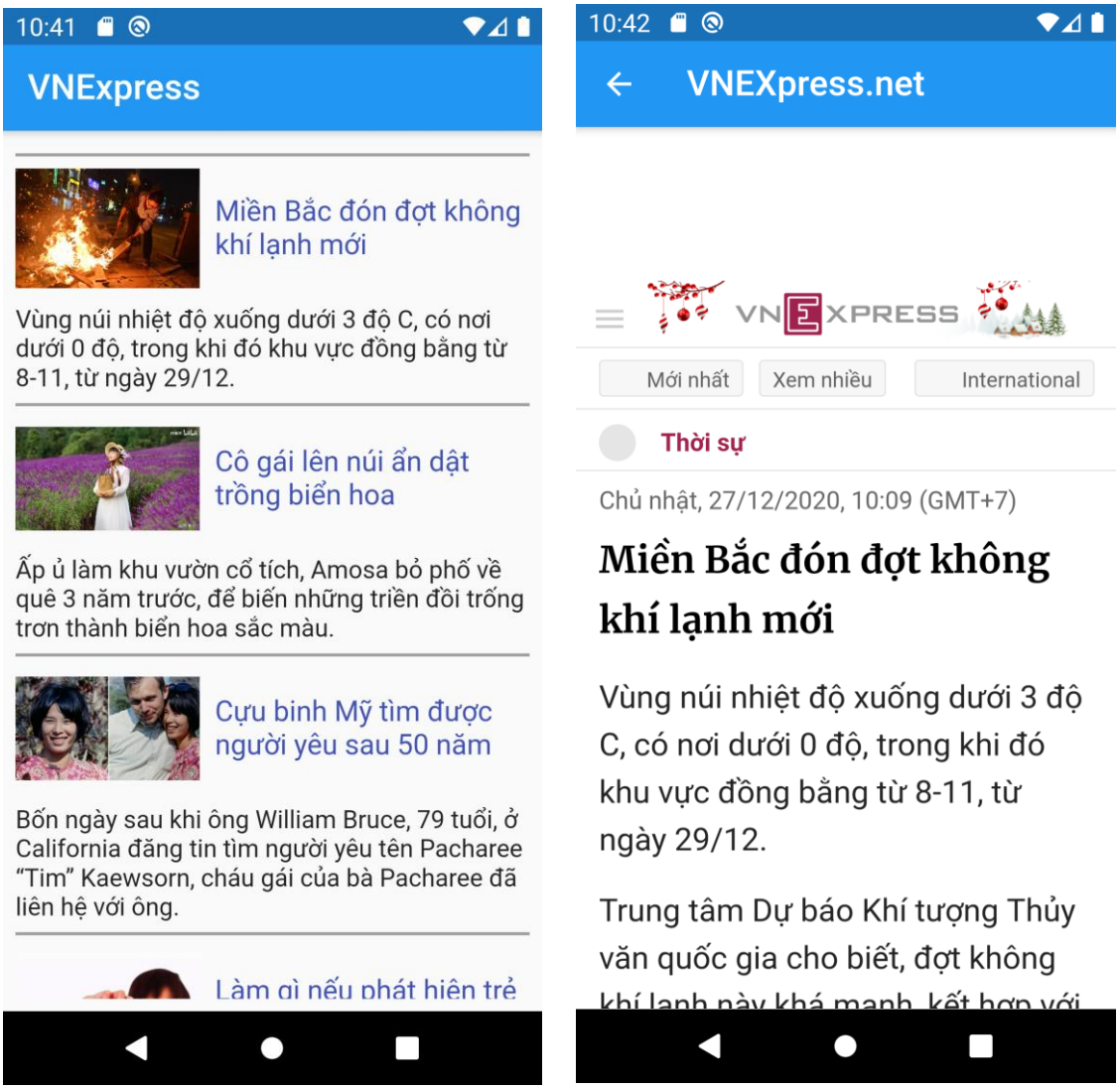

9:54
Môn học
CTDL
Số tín chỉ: 3
LTC
Số tín chỉ: 3
LTTDD
Số tín chỉ: 3
9:59
Môn học
Xem
Xóa
CTDL
Số tín chỉ: 3
LTC
Số tín chỉ: 3
LTTDD
Số tín chỉ: 3


10:01
Điểm môn học CTDL
Kiem Tra CTDL 1
Điểm: 6.0
Hệ số: 2.0
Kiem Tra CTDL 2
Điểm: 7.0
Hệ số: 3.0


10:02
Điểm môn học CTDL
Xem
Xóa
Kiem Tra CTI
Điểm: 6.0
Hệ số: 2.0
Kiem Tra CTDL 2
Điểm: 7.0
Hệ số: 3.0



Bài tập tổng hợp 1: 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 hiện thị nội dụng 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) =>
value.listRSSItems);
    return Scaffold(
      appBar: AppBar(
        title: Text('VNExpress'),
      ),
      body: buildListRSS(context, rssItems),
    );
  }

  Widget buildListRSS(BuildContext context, List<RSSItem> rssItems) {
    if(rssItems.isEmpty)
```

```
      return Center(child: CircularProgressIndicator());
    return RefreshIndicator(
      onRefresh: () {
        RSSProvider provider = context.read<RSSProvider>();
        return provider.readRSSItems();
      },

      child: Padding(
        padding: const EdgeInsets.all(8.0),
        child: ListView.separated(
            itemCount: rssItems.length,
            separatorBuilder: (context, index) => Divider(
              color: Colors.grey,
              thickness: 2,
            ),
            itemBuilder:(context, index) =>  Column(
              children: [
                GestureDetector(
                  onTap: () => Navigator.push(
                    context,
                    MaterialPageRoute(builder: (context) => MyWebPage(url:
rssItems[index].link,),)
                  ),
                  child: Row(
                    children: [
                      Container(
                        height: 80,
                        width: 120,
                        child: _getImage(rssItems[index].imageUrl),
                      ),
                      SizedBox(width: 10,),
                      Expanded(
                        child: Text(
                          rssItems[index].title,
                          style: TextStyle(fontSize: 16,
                            color: Colors.indigo,
                          ),
                        ),
                      ),
                    ],
                  ),
                ),
                SizedBox(height: 10,),
                Text(
                    rssItems[index].description,
                  style: TextStyle(fontFamily: "Roboto"),
                ),
                //Text("Hiển thị được tiếng Việt rồi")
              ],
            )
        ),
      ),
    );

  }

  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 2: Viết ứng dụng quản lý chi tiêu đơn giản trên Flutter

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