**Computer Science Internal Assessment:** **French Vocabulary Memorization Software**

APPENDIX B: SOURCE CODE

# **File Structure:**

* *pages* contains all the screens that make up the software
* *services* contains the file operations system for the software
* *main.dart* contains the routing system for the software
* *pubspec.yaml* lists the software’s dependencies and metadata

Structure of Source Code:

French\_Vocabulary\_Memorization\_Software

pages

home.dart

battlemode.dart

bmeasy.dart

bmhard.dart

bmmedium.dart

easytest.dart

eta.dart

test.dart

testA.dart

hardtest.dart

hta.dart

score.dart

scoreE.dart

scoreH.dart

sa.dart

esa.dart

hsa.dart

progress.dart

progressE.dart

progressH.dart

pa.dart

pea.dart

hpa.dart

services

F\_O.dart

main.dart

pubspec.yaml

# **Pages:**

## ***home.dart:***

import 'package:flutter/material.dart';  
import 'package:french\_learning\_ia/pages/easytest.dart';  
import 'package:french\_learning\_ia/pages/eta.dart';  
import 'package:french\_learning\_ia/pages/hardtest.dart';  
import 'package:french\_learning\_ia/pages/hta.dart';  
import 'package:french\_learning\_ia/pages/test.dart';  
import 'package:french\_learning\_ia/pages/testA.dart';  
class Home extends StatefulWidget {  
 @override  
 \_HomeState createState() => \_HomeState();  
}  
const sep = 10.0;  
const buttonsep = 15.0;  
class \_HomeState extends State<Home> {  
 @override  
 Widget build(BuildContext context) {  
 return Scaffold(  
 backgroundColor: Colors.*orange*[400],  
 appBar: AppBar(  
 title: Text(  
 'Home Screen',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 22.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 backgroundColor: Colors.*deepOrange*,  
 centerTitle: true,  
 ),  
 body: Column(  
 children: [  
 Center(  
 child:Padding(  
 padding: const EdgeInsets.fromLTRB(0,20,0,sep),  
 child: Text(  
 'French To English:',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 18.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 ),  
 ),  
 Center(  
 child: Padding(  
 padding: const EdgeInsets.fromLTRB(0.0, 0.0, 0.0, buttonsep),  
 child: ButtonTheme(  
 minWidth: 245.0,  
 height: 30.0,  
 child: RaisedButton(  
 color: Colors.*white*,  
 child: Text(  
 'Take A New Easy Level Test.',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 19.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 onPressed: (){  
 Navigator.*pushReplacementNamed*(context, '/et');  
 wordE = currentTestE.questionList[0];  
 expectedAnswerE = currentTestE.answersList[0];  
 },  
 ),  
 ),  
 ),  
 ),  
 Center(  
 child: Padding(  
 padding: const EdgeInsets.fromLTRB(0.0, 0.0, 0.0, buttonsep),  
 child: ButtonTheme(  
 minWidth: 245.0,  
 height: 30.0,  
 child: RaisedButton(  
 color: Colors.*white*,  
 child: Text(  
 'Take A New Medium Level Test.',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 19.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 onPressed: (){  
 Navigator.*pushReplacementNamed*(context, '/test');  
 setState(() {  
 word = currentTest.questionList[0];  
 expectedAnswer = currentTest.answersList[0];  
 });  
 },  
 ),  
 ),  
 ),  
 ),  
 Center(  
 child: Padding(  
 padding: const EdgeInsets.fromLTRB(0,0,0,sep),  
 child: ButtonTheme(  
 minWidth: 245.0,  
 height: 30.0,  
 child: RaisedButton(  
 color: Colors.*white*,  
 child: Text(  
 'Take A New Hard Level Test.',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 19.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 onPressed: (){  
 wordH = currentTestH.questionList[0];  
 expectedAnswerH = currentTestH.answersList[0];  
 Navigator.*pushReplacementNamed*(context, '/ht');  
 },  
 ),  
 ),  
 ),  
 ),  
 Center(  
 child:Padding(  
 padding: const EdgeInsets.fromLTRB(0,0,0,sep),  
 child: Text(  
 'English To French:',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 19.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 ),  
 ),  
 Center(  
 child: Padding(  
 padding: const EdgeInsets.fromLTRB(0.0, 0.0, 0.0, buttonsep),  
 child: ButtonTheme(  
 minWidth: 245.0,  
 height: 30.0,  
 child: RaisedButton(  
 color: Colors.*white*,  
 child: Text(  
 'Take A New Easy Level Test.',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 19.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 onPressed: (){  
 wordEA = currentTestEA.questionList[0];  
 expectedAnswerEA = currentTestEA.answersList[0];  
 Navigator.*pushReplacementNamed*(context, '/eta');  
 },  
 ),  
 ),  
 ),  
 ),  
 Center(  
 child: Padding(  
 padding: const EdgeInsets.fromLTRB(0.0, 0.0, 0.0, buttonsep),  
 child: ButtonTheme(  
 minWidth: 245.0,  
 height: 30.0,  
 child: RaisedButton(  
 color: Colors.*white*,  
 child: Text(  
 'Take A New Medium Level Test.',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 19.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 onPressed: (){  
 Navigator.*pushReplacementNamed*(context, '/testA');  
 wordA = currentTestA.questionList[0];  
 expectedAnswerA = currentTestA.answersList[0];  
 },  
 ),  
 ),  
 ),  
 ),  
 Center(  
 child: Padding(  
 padding: const EdgeInsets.fromLTRB(0,0,0,sep),  
 child: ButtonTheme(  
 minWidth: 245.0,  
 height: 30.0,  
 child: RaisedButton(  
 color: Colors.*white*,  
 child: Text(  
 'Take A New Hard Level Test.',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 19.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 onPressed: (){  
 wordHA = currentTestHA.questionList[0];  
 expectedAnswerHA = currentTestHA.answersList[0];  
 Navigator.*pushReplacementNamed*(context, '/hta');  
 },  
 ),  
 ),  
 ),  
 ),  
 Center(  
 child:Padding(  
 padding: const EdgeInsets.fromLTRB(0,0,0,sep),  
 child: Text(  
 'Battle Mode!',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 19.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 ),  
 ),  
 Center(  
 child: ButtonTheme(  
 minWidth: 245.0,  
 height: 30.0,  
 child: Padding(  
 padding: const EdgeInsets.fromLTRB(0,0,0,sep),  
 child: RaisedButton(  
 color: Colors.*white*,  
 child: Text(  
 'Play Battle Mode!',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 19.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 onPressed: (){  
 Navigator.*pushReplacementNamed*(context, '/bm');  
 },  
 ),  
 ),  
 ),  
 ),  
 ],  
 ),  
 );  
 }  
}

## ***battlemode.dart:***

import 'package:flutter/material.dart';  
  
  
  
class BattleMode extends StatefulWidget {  
 @override  
 \_BattleModeState createState() => \_BattleModeState();  
}  
  
const sep = 20.0;  
const wid = 235.0;  
class \_BattleModeState extends State<BattleMode> {  
 @override  
 Widget build(BuildContext context) {  
 return Scaffold(  
 backgroundColor: Colors.*orange*[400],  
 appBar: AppBar(  
 title: Text(  
 'Battle Mode',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 22.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 leading: IconButton(  
 onPressed: () {  
 Navigator.*pushReplacementNamed*(context, '/');  
 },  
 icon: Icon(Icons.*arrow\_back*),  
 ),  
 backgroundColor: Colors.*deepOrange*,  
 centerTitle: true,  
 ),  
 body: Column(  
 children: [  
 Center(  
 child: Padding(  
 padding: const EdgeInsets.fromLTRB(0.0, sep, 0, 0),  
 child: Center(  
 child: ButtonTheme(  
 minWidth: 245.0,  
 height: 30.0,  
 child: RaisedButton(  
 color: Colors.*white*,  
 child: Text(  
 'Easy Difficulty Mode!',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 18.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 onPressed: (){  
 Navigator.*pushReplacementNamed*(context, '/bme');  
 },  
 ),  
 ),  
 ),  
 ),  
 ),  
 Center(  
 child: Padding(  
 padding: const EdgeInsets.fromLTRB(0.0, sep, 0, 0),  
 child: Center(  
 child: ButtonTheme(  
 minWidth: 245.0,  
 height: 30.0,  
 child: RaisedButton(  
 color: Colors.*white*,  
 child: Text(  
 'Medium Difficulty Mode!',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 18.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 onPressed: (){  
 Navigator.*pushReplacementNamed*(context, '/bmm');  
 },  
 ),  
 ),  
 ),  
 ),  
 ),  
 Center(  
 child: Padding(  
 padding: const EdgeInsets.fromLTRB(0.0, sep, 0, 0),  
 child: Center(  
 child: ButtonTheme(  
 minWidth: 245.0,  
 height: 30.0,  
 child: RaisedButton(  
 color: Colors.*white*,  
 child: Text(  
 'Hard Difficulty Mode!',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 18.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 onPressed: (){  
 Navigator.*pushReplacementNamed*(context, '/bmh');  
 },  
 ),  
 ),  
 ),  
 ),  
 ),  
 ],  
 ),  
 );  
 }  
}

## ***bmeasy.dart:***

import 'package:flutter/material.dart';  
import 'dart:math';  
  
import 'package:french\_learning\_ia/pages/test.dart';  
  
var question = new Random ();  
var noCorrect = 0;  
var isSoftwareCorrect;  
var noSoftwareCorrect = 0;  
var displaynoSoftwareCorrect = noSoftwareCorrect.toString();  
var counter = 0;  
var accuracyDeterminer = new Random ();  
var accuracyNumber = accuracyDeterminer.nextInt(20);  
var element = question.nextInt(noQuestions);  
var previous;  
var expectedAnswer = answers.elementAt(element);  
var howGood = 9;  
var answers = ['etre','avoir','pouvoir','faire','mettre','dire','devoir','prendre','donner','aller','vouloir','savoir','etre','falloir','lire','finir','manger','prendre','regarder', 'voir', 'venir', 'suivre', 'parler', 'croire', 'aimer', 'passer', 'penser', 'attendre', 'trouver', 'laisser', 'arriver', 'donner', 'appeler', 'partir', 'mettre', 'rester', 'arreter', 'connaitre', 'demander', 'comprendre', 'sortir', 'entendre', 'chercher', 'aider', 'essayer', 'revenir', 'jouer', 'finir', 'perdre', 'sentir','to be','to have','to be able to','to do','to put','to say','to have to','to take','to give','to go','to want','to know','to be','to have to','to read','to finish','to eat','to take','to look', 'to see', 'to come', 'to follow', 'to talk', 'to believe', 'to like', 'to pass', 'to think', 'to wait for', 'to find', 'to leave', 'to arrive', 'to give', 'to call', 'to go', 'to put', 'to stay', ' to stop', 'to know','to ask', 'to understand', 'to go out', 'to hear', 'to look for', 'to help', 'to try', 'to come back', 'to play', 'to finish', 'to lose', 'to smell',];  
var questions = ['to be','to have','to be able to','to do','to put','to say','to have to','to take','to give','to go','to want','to know','to be','to have to','to read','to finish','to eat','to take','to look', 'to see', 'to come', 'to follow', 'to talk', 'to believe', 'to like', 'to pass', 'to think', 'to wait for', 'to find', 'to leave', 'to arrive', 'to give', 'to call', 'to go', 'to put', 'to stay', ' to stop', 'to know','to ask', 'to understand', 'to go out', 'to hear', 'to look for', 'to help', 'to try', 'to come back', 'to play', 'to finish', 'to lose', 'to smell','etre','avoir','pouvoir','faire','mettre','dire','devoir','prendre','donner','aller','vouloir','savoir','etre','falloir','lire','finir','manger','prendre','regarder', 'voir', 'venir', 'suivre', 'parler', 'croire', 'aimer', 'passer', 'penser', 'attendre', 'trouver', 'laisser', 'arriver', 'donner', 'appeler', 'partir', 'mettre', 'rester', 'arreter', 'connaitre', 'demander', 'comprendre', 'sortir', 'entendre', 'chercher', 'aider', 'essayer', 'revenir', 'jouer', 'finir', 'perdre', 'sentir',];  
var isCorrect = false;  
var answerMessage = "Waiting for response....";  
var noQuestions = 100;  
var displayNoCorrect = noCorrect.toString();  
var winnerStatus = 'Waiting to start battle...';  
var displayNumberQ = counter.toString();  
var SoftwareAnswer = "Waiting for answer...";  
var wrong;  
  
class EasyBM extends StatefulWidget {  
 @override  
 \_EasyBMState createState() => \_EasyBMState();  
 void initState () {  
  
 }  
}  
  
class \_EasyBMState extends State<EasyBM> {  
 String word = questions.elementAt(element);  
 String answer;  
 final myController = TextEditingController();  
 @override  
 void dispose() {  
 // Clean up the controller when the widget is disposed.  
 myController.dispose();  
 super.dispose();  
 }  
 Widget build(BuildContext context) {  
 return Scaffold(  
 backgroundColor: Colors.*orange*[400],  
 appBar: AppBar(  
 title: Text(  
 'Battle Mode - Easy',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 22.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 leading: IconButton(  
 onPressed: () {  
 Navigator.*pushReplacementNamed*(context, '/bm');  
 },  
 icon: Icon(Icons.*arrow\_back*),  
 ),  
 backgroundColor: Colors.*deepOrange*,  
 centerTitle: true,  
 ),  
 body: SingleChildScrollView(  
 child: Column(  
 children: <Widget>[  
 Padding(  
 padding: const EdgeInsets.fromLTRB(0.0, 10.0, 0.0, 0.0),  
 child: Center(  
 child: Text(  
 "Question : What is the equivalent of '$word' ?",  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 20.0,  
 fontWeight: FontWeight.*bold*,  
 color: Colors.*black*,  
 ),  
 ),  
 ),  
 ),  
 Padding(  
 padding: const EdgeInsets.fromLTRB(15.0, 30.0, 0.0, 0.0),  
 child: Center(  
 child: Text(  
 'Answer:',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 20.0,  
 fontWeight: FontWeight.*bold*,  
 color: Colors.*black*,  
 ),  
 ),  
 ),  
 ),  
 TextField(  
 controller: myController,  
 textAlign: TextAlign.center,  
 decoration: InputDecoration(  
 contentPadding: EdgeInsets.*zero*,  
 hintText: "Enter Answer Here...",  
 ),  
 onSubmitted: (answer){  
 },  
 ),  
 SizedBox(height: 50.0),  
 Padding(  
 padding: const EdgeInsets.fromLTRB(0.0, 10.0, 0.0, 0.0),  
 child: Center(  
 child: Text(  
 "Answer Status : $answerMessage",  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 18.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 ),  
 ),  
 Center(  
 child: Padding(  
 padding: const EdgeInsets.fromLTRB(0.0, 10.0, 0.0, 0.0),  
 child: Text(  
 "Player Score: $displayNoCorrect",  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 18.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 ),  
 ),  
 Center(  
 child: Padding(  
 padding: const EdgeInsets.fromLTRB(0.0, 10.0, 0.0, 0.0),  
 child: Text(  
 "Software Answer: $SoftwareAnswer",  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 18.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 ),  
 ),  
 Center(  
 child: Padding(  
 padding: const EdgeInsets.fromLTRB(0.0, 10.0, 0.0, 0.0),  
 child: Text(  
 "Software Score: $displaynoSoftwareCorrect",  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 18.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 ),  
 ),  
 Center(  
 child: Padding(  
 padding: const EdgeInsets.fromLTRB(0.0, 10.0, 0.0, 0.0),  
 child: Text(  
 "Winning Status: $winnerStatus",  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 18.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 ),  
 ),  
 Center(  
 child: Padding(  
 padding: const EdgeInsets.fromLTRB(0.0, 10.0, 0.0, 0.0),  
 child: Text(  
 "Questions Answered: $displayNumberQ",  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 18.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 ),  
 ),  
 Padding(  
 padding: const EdgeInsets.fromLTRB(0.0, 10.0, 0.0, 0.0),  
 child: Center(  
 child: ButtonTheme(  
 minWidth: 200.0,  
 height: 30.0,  
 child: RaisedButton(  
 color: Colors.*white*,  
 child: Text(  
 'Check Answer.',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 18.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 onPressed: (){  
 answer = myController.text;  
 isCorrect = (answer == expectedAnswer);  
 accuracyNumber = accuracyDeterminer.nextInt(20);  
 if (accuracyNumber >= 0 && accuracyNumber <= howGood) {  
 setState(() {  
 isSoftwareCorrect = true;  
 });  
 }  
 if (accuracyNumber >= (howGood+1) && accuracyNumber <= 20) {  
 setState(() {  
 isSoftwareCorrect = false;  
 });  
 }  
 answerMessage = (isCorrect == true ) ? "Correct!" : "Sorry, that answer was incorrect.\n The expected answer to this question was $expectedAnswer.";  
 if (isCorrect == true) {  
 setState(() {  
 noCorrect = noCorrect + 1;  
 displayNoCorrect = noCorrect.toString();  
 });  
 }  
 if (isSoftwareCorrect == true) {  
 setState(() {  
 noSoftwareCorrect = noSoftwareCorrect + 1;  
 displaynoSoftwareCorrect = noSoftwareCorrect.toString();  
 SoftwareAnswer = currentTest.answersList[element];  
 });  
 }  
 else if (isSoftwareCorrect == false) {  
 setState(() {  
 do {  
 wrong = question.nextInt(noQuestions);  
 } while (wrong == element);  
 SoftwareAnswer = answers[wrong];  
 });  
 }  
 if (noSoftwareCorrect == noCorrect) {  
 setState(() {  
 winnerStatus = 'Draw';  
 });  
 }  
 if (noSoftwareCorrect > noCorrect) {  
 setState(() {  
 winnerStatus = 'You are losing!';  
 });  
 }  
 if (noSoftwareCorrect < noCorrect) {  
 setState(() {  
 winnerStatus = 'You are winning!';  
 });  
 }  
 Future.delayed(const Duration(milliseconds: 2500), () {  
 setState(() {  
 previous = element;  
 do {  
 element = question.nextInt(noQuestions);  
 } while (previous == element);  
 expectedAnswer = answers.elementAt(element);  
 word = currentTest.questionList[element];  
 answerMessage = "Waiting for response....";  
 SoftwareAnswer = "Waiting for answer...";  
 });  
 });  
 setState(() {  
 counter = counter + 1;  
 displayNumberQ = counter.toString();  
 });  
 },  
 ),  
 ),  
 ),  
 ),  
 Center(  
 child: Padding(  
 padding: const EdgeInsets.fromLTRB(0.0, 10.0, 0.0, 0.0),  
 child: ButtonTheme(  
 minWidth: 200.0,  
 height: 30.0,  
 child: RaisedButton(  
 color: Colors.*white*,  
 child: Text(  
 'Return to Home.',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 18.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 onPressed: (){  
 setState(() {  
 noSoftwareCorrect = 0;  
 noCorrect = 0;  
 counter = 0;  
 answerMessage = "Waiting for response....";  
 winnerStatus = 'Waiting to start battle...';  
 displaynoSoftwareCorrect = noSoftwareCorrect.toString();  
 displayNoCorrect = noCorrect.toString();  
 displayNumberQ = counter.toString();  
 });  
 Navigator.*pushReplacementNamed*(context, '/');  
 }  
 ),  
 ),  
 ),  
 ),  
 ],  
 ),  
 ),  
 );  
 }  
}

## ***bmhard.dart:***

import 'package:flutter/material.dart';  
import 'dart:math';  
  
var question = new Random ();  
var noCorrect = 0;  
var isSoftwareCorrect;  
var noSoftwareCorrect = 0;  
var displaynoSoftwareCorrect = noSoftwareCorrect.toString();  
var counter = 0;  
var accuracyDeterminer = new Random ();  
var accuracyNumber = accuracyDeterminer.nextInt(20);  
var element = question.nextInt(noQuestions);  
var previous;  
var expectedAnswer = answers.elementAt(element);  
var howGood = 18;  
var questions = ['adorer','recevoir','utiliser','coucher','preferer','offrir','preparer','choisir','conduire','chanter','presenter','accepter','refuser', 'terminer','amuser','intéresser','rire','pardonner','embrasser','danser','detester','maintenir','supposer','epouser','approcher','craindre','crier','inviter','arranger','remercier','repeter','signer','accompagner','oser','permettre','annuler','laver','se plaindre','epeler','traduire','eteindre','allumer','se réveiller','ajouter','gouter','couter','cuire','décrire', 'effacer', 'enseigner','to adore','to receive','to use','to put to bed','to prefer','to offer','to prepare','to choose','to drive','to sing','to introduce','to accept','to refuse', 'to finish','to amuse','to interest','to laugh','to forgive','to kiss','to dance','to hate','to keep','to suppose','to marry','to approach','to fear','to shout','to invite','to arrange','to thank','to repeat','to sign','to go with','to dare','to allow','to cancel','to clean','to complain','to spell', 'to translate', 'to turn off','to turn on','to wake up','to add', 'to taste', 'to cost', 'to cook','to describe', 'to erase', 'to teach',];  
var answers = ['to adore','to receive','to use','to put to bed','to prefer','to offer','to prepare','to choose','to drive','to sing','to introduce','to accept','to refuse', 'to finish','to amuse','to interest','to laugh','to forgive','to kiss','to dance','to hate','to keep','to suppose','to marry','to approach','to fear','to shout','to invite','to arrange','to thank','to repeat','to sign','to go with','to dare','to allow','to cancel','to clean','to complain','to spell', 'to translate', 'to turn off','to turn on','to wake up','to add', 'to taste', 'to cost', 'to cook','to describe', 'to erase', 'to teach','adorer','recevoir','utiliser','coucher','preferer','offrir','preparer','choisir','conduire','chanter','presenter','accepter','refuser', 'terminer','amuser','intéresser','rire','pardonner','embrasser','danser','detester','maintenir','supposer','epouser','approcher','craindre','crier','inviter','arranger','remercier','repeter','signer','accompagner','oser','permettre','annuler','laver','se plaindre','epeler','traduire','eteindre','allumer','se réveiller','ajouter','gouter','couter','cuire','décrire', 'effacer', 'enseigner',];  
var isCorrect = false;  
var answerMessage = "Waiting for response....";  
var noQuestions = 100;  
var displayNoCorrect = noCorrect.toString();  
var winnerStatus = 'Waiting to start battle...';  
var displayNumberQ = counter.toString();  
class HardBM extends StatefulWidget {  
 @override  
 \_HardBMState createState() => \_HardBMState();  
 void initState () {  
  
 }  
}  
  
class \_HardBMState extends State<HardBM> {  
 String word = questions.elementAt(element);  
 String answer;  
 final myController = TextEditingController();  
 @override  
 void dispose() {  
 // Clean up the controller when the widget is disposed.  
 myController.dispose();  
 super.dispose();  
 }  
 Widget build(BuildContext context) {  
 return Scaffold(  
 backgroundColor: Colors.*orange*[400],  
 appBar: AppBar(  
 title: Text(  
 'Battle Mode - Hard',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 22.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 leading: IconButton(  
 onPressed: () {  
 Navigator.*pushReplacementNamed*(context, '/bm');  
 },  
 icon: Icon(Icons.*arrow\_back*),  
 ),  
 backgroundColor: Colors.*deepOrange*,  
 centerTitle: true,  
 ),  
 body: SingleChildScrollView(  
 child: Column(  
 children: <Widget>[  
 Padding(  
 padding: const EdgeInsets.fromLTRB(0.0, 10.0, 0.0, 0.0),  
 child: Center(  
 child: Text(  
 "Question : What is the equivalent of '$word' ?",  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 20.0,  
 fontWeight: FontWeight.*bold*,  
 color: Colors.*black*,  
 ),  
 ),  
 ),  
 ),  
 Padding(  
 padding: const EdgeInsets.fromLTRB(15.0, 30.0, 0.0, 0.0),  
 child: Center(  
 child: Text(  
 'Answer:',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 20.0,  
 fontWeight: FontWeight.*bold*,  
 color: Colors.*black*,  
 ),  
 ),  
 ),  
 ),  
 TextField(  
 controller: myController,  
 textAlign: TextAlign.center,  
 decoration: InputDecoration(  
 contentPadding: EdgeInsets.*zero*,  
 hintText: "Enter Answer Here...",  
 ),  
 onSubmitted: (answer){  
 },  
 ),  
 SizedBox(height: 50.0),  
 Padding(  
 padding: const EdgeInsets.fromLTRB(0.0, 10.0, 0.0, 0.0),  
 child: Center(  
 child: Text(  
 "Answer Status : $answerMessage",  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 18.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 ),  
 ),  
 Center(  
 child: Padding(  
 padding: const EdgeInsets.fromLTRB(0.0, 10.0, 0.0, 0.0),  
 child: Text(  
 "Player Score: $displayNoCorrect",  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 18.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 ),  
 ),  
 Center(  
 child: Padding(  
 padding: const EdgeInsets.fromLTRB(0.0, 10.0, 0.0, 0.0),  
 child: Text(  
 "Software Score: $displaynoSoftwareCorrect",  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 18.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 ),  
 ),  
 Center(  
 child: Padding(  
 padding: const EdgeInsets.fromLTRB(0.0, 10.0, 0.0, 0.0),  
 child: Text(  
 "Winning Status: $winnerStatus",  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 18.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 ),  
 ),  
 Center(  
 child: Padding(  
 padding: const EdgeInsets.fromLTRB(0.0, 10.0, 0.0, 0.0),  
 child: Text(  
 "Questions Answered: $displayNumberQ",  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 18.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 ),  
 ),  
 Padding(  
 padding: const EdgeInsets.fromLTRB(0.0, 10.0, 0.0, 0.0),  
 child: Center(  
 child: ButtonTheme(  
 minWidth: 200.0,  
 height: 30.0,  
 child: RaisedButton(  
 color: Colors.*white*,  
 child: Text(  
 'Check Answer.',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 18.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 onPressed: (){  
 answer = myController.text;  
 isCorrect = (answer == expectedAnswer);  
 accuracyNumber = accuracyDeterminer.nextInt(20);  
 if (accuracyNumber >= 0 && accuracyNumber <= howGood) {  
 setState(() {  
 isSoftwareCorrect = true;  
 });  
 }  
 if (accuracyNumber >= (howGood+1) && accuracyNumber <= 20) {  
 setState(() {  
 isSoftwareCorrect = false;  
 });  
 }  
 answerMessage = (isCorrect == true ) ? "Correct!" : "Sorry, that answer was incorrect.\n The expected answer to this question was $expectedAnswer.";  
 if (isCorrect == true) {  
 setState(() {  
 noCorrect = noCorrect + 1;  
 displayNoCorrect = noCorrect.toString();  
 });  
 }  
 if (isSoftwareCorrect == true) {  
 setState(() {  
 noSoftwareCorrect = noSoftwareCorrect + 1;  
 displaynoSoftwareCorrect = noSoftwareCorrect.toString();  
 });  
 }  
 if (noSoftwareCorrect == noCorrect) {  
 setState(() {  
 winnerStatus = 'Draw';  
 });  
 }  
 if (noSoftwareCorrect > noCorrect) {  
 setState(() {  
 winnerStatus = 'You are losing!';  
 });  
 }  
 if (noSoftwareCorrect < noCorrect) {  
 setState(() {  
 winnerStatus = 'You are winning!';  
 });  
 }  
 Future.delayed(const Duration(milliseconds: 2500), () {  
 setState(() {  
 previous = element;  
 do {  
 element = question.nextInt(noQuestions);  
 } while (previous == element);  
 expectedAnswer = answers.elementAt(element);  
 word = questions.elementAt(element);  
 answerMessage = "Waiting for response....";  
 });  
 });  
 setState(() {  
 counter = counter + 1;  
 displayNumberQ = counter.toString();  
 });  
 },  
 ),  
 ),  
 ),  
 ),  
 Center(  
 child: Padding(  
 padding: const EdgeInsets.fromLTRB(0.0, 10.0, 0.0, 0.0),  
 child: ButtonTheme(  
 minWidth: 200.0,  
 height: 30.0,  
 child: RaisedButton(  
 color: Colors.*white*,  
 child: Text(  
 'Return to Home.',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 18.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 onPressed: (){  
 setState(() {  
 noSoftwareCorrect = 0;  
 noCorrect = 0;  
 counter = 0;  
 answerMessage = "Waiting for response....";  
 winnerStatus = 'Waiting to start battle...';  
 displaynoSoftwareCorrect = noSoftwareCorrect.toString();  
 displayNoCorrect = noCorrect.toString();  
 displayNumberQ = counter.toString();  
 });  
  
 Navigator.*pushReplacementNamed*(context, '/');  
  
 }  
 ),  
 ),  
 ),  
 ),  
 ],  
 ),  
 ),  
 );  
 }  
}

## ***bmmedium.dart:***

import 'package:flutter/material.dart';  
import 'dart:math';  
  
var question = new Random ();  
var noCorrect = 0;  
var isSoftwareCorrect;  
var noSoftwareCorrect = 0;  
var displaynoSoftwareCorrect = noSoftwareCorrect.toString();  
var counter = 0;  
var accuracyDeterminer = new Random ();  
var accuracyNumber = accuracyDeterminer.nextInt(20);  
var element = question.nextInt(noQuestions);  
var previous;  
var expectedAnswer = answers.elementAt(element);  
var howGood = 14;  
var questions = ['rentrer', 'vivre', 'rendre', 'tenir', 'oublier', 'travailler', 'manger', 'entrer', 'devenir', 'commencer', 'payer', 'tirer', 'ouvrir', 'changer', 'excuser', 'dormir', 'occuper', 'marcher', 'envoyer', 'apprendre', 'boire', 'garder', 'montrer', 'asseoir', 'porter', 'prier', 'servir', 'écrire', 'retrouver', 'gagner', 'acheter', 'rappeler', 'lire', 'monter', 'quitter', 'emmener', 'toucher', 'continuer', 'raconter', 'repondre', 'sauver', 'rencontrer', 'fermer', 'valoir', 'compter', 'bouger', 'apporter', 'décider', 'vendre', 'expliquer','to bring in','to live','to return','to hold','to forget','to work','to eat','to go in','to become','to start','to pay','to pull','to open','to change','to forgive','to sleep','to occupy','to walk','to send','to learn','to drink','to look after','to show','to sit down','to carry','to pray','to serve','to write','to find', 'to win', 'to buy', 'to remind', 'to read', 'to go up', 'to leave', 'to take', 'to touch', 'to continue', 'to tell', 'to answer', 'to save', 'to meet', 'to close', 'to hold', 'to count', 'to move', 'to bring', 'to decide', 'to sell', 'to explain',];  
var answers = ['to bring in','to live','to return','to hold','to forget','to work','to eat','to go in','to become','to start','to pay','to pull','to open','to change','to forgive','to sleep','to occupy','to walk','to send','to learn','to drink','to look after','to show','to sit down','to carry','to pray','to serve','to write','to find', 'to win', 'to buy', 'to remind', 'to read', 'to go up', 'to leave', 'to take', 'to touch', 'to continue', 'to tell', 'to answer', 'to save', 'to meet', 'to close', 'to hold', 'to count', 'to move', 'to bring', 'to decide', 'to sell', 'to explain','rentrer', 'vivre', 'rendre', 'tenir', 'oublier', 'travailler', 'manger', 'entrer', 'devenir', 'commencer', 'payer', 'tirer', 'ouvrir', 'changer', 'excuser', 'dormir', 'occuper', 'marcher', 'envoyer', 'apprendre', 'boire', 'garder', 'montrer', 'asseoir', 'porter', 'prier', 'servir', 'écrire', 'retrouver', 'gagner', 'acheter', 'rappeler', 'lire', 'monter', 'quitter', 'emmener', 'toucher', 'continuer', 'raconter', 'repondre', 'sauver', 'rencontrer', 'fermer', 'valoir', 'compter', 'bouger', 'apporter', 'décider', 'vendre', 'expliquer',];  
var isCorrect = false;  
var answerMessage = "Waiting for response....";  
var noQuestions = 100;  
var displayNoCorrect = noCorrect.toString();  
var winnerStatus = 'Waiting to start battle...';  
var displayNumberQ = counter.toString();  
class MediumBM extends StatefulWidget {  
 @override  
 \_MediumBMState createState() => \_MediumBMState();  
 void initState () {  
  
 }  
}  
  
class \_MediumBMState extends State<MediumBM> {  
 String word = questions.elementAt(element);  
 String answer;  
 final myController = TextEditingController();  
 @override  
 void dispose() {  
 // Clean up the controller when the widget is disposed.  
 myController.dispose();  
 super.dispose();  
 }  
 Widget build(BuildContext context) {  
 return Scaffold(  
 backgroundColor: Colors.*orange*[400],  
 appBar: AppBar(  
 title: Text(  
 'Battle Mode - Medium',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 22.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 leading: IconButton(  
 onPressed: () {  
 Navigator.*pushReplacementNamed*(context, '/bm');  
 },  
 icon: Icon(Icons.*arrow\_back*),  
 ),  
 backgroundColor: Colors.*deepOrange*,  
 centerTitle: true,  
 ),  
 body: SingleChildScrollView(  
 child: Column(  
 children: <Widget>[  
 Padding(  
 padding: const EdgeInsets.fromLTRB(0.0, 10.0, 0.0, 0.0),  
 child: Center(  
 child: Text(  
 "Question : What is the equivalent of '$word' ?",  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 20.0,  
 fontWeight: FontWeight.*bold*,  
 color: Colors.*black*,  
 ),  
 ),  
 ),  
 ),  
 Padding(  
 padding: const EdgeInsets.fromLTRB(15.0, 30.0, 0.0, 0.0),  
 child: Center(  
 child: Text(  
 'Answer:',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 20.0,  
 fontWeight: FontWeight.*bold*,  
 color: Colors.*black*,  
 ),  
 ),  
 ),  
 ),  
 TextField(  
 controller: myController,  
 textAlign: TextAlign.center,  
 decoration: InputDecoration(  
 contentPadding: EdgeInsets.*zero*,  
 hintText: "Enter Answer Here...",  
 ),  
 onSubmitted: (answer){  
 },  
 ),  
 SizedBox(height: 50.0),  
 Padding(  
 padding: const EdgeInsets.fromLTRB(0.0, 10.0, 0.0, 0.0),  
 child: Center(  
 child: Text(  
 "Answer Status : $answerMessage",  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 18.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 ),  
 ),  
 Center(  
 child: Padding(  
 padding: const EdgeInsets.fromLTRB(0.0, 10.0, 0.0, 0.0),  
 child: Text(  
 "Player Score: $displayNoCorrect",  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 18.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 ),  
 ),  
 Center(  
 child: Padding(  
 padding: const EdgeInsets.fromLTRB(0.0, 10.0, 0.0, 0.0),  
 child: Text(  
 "Software Score: $displaynoSoftwareCorrect",  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 18.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 ),  
 ),  
 Center(  
 child: Padding(  
 padding: const EdgeInsets.fromLTRB(0.0, 10.0, 0.0, 0.0),  
 child: Text(  
 "Winning Status: $winnerStatus",  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 18.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 ),  
 ),  
 Center(  
 child: Padding(  
 padding: const EdgeInsets.fromLTRB(0.0, 10.0, 0.0, 0.0),  
 child: Text(  
 "Questions Answered: $displayNumberQ",  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 18.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 ),  
 ),  
 Padding(  
 padding: const EdgeInsets.fromLTRB(0.0, 10.0, 0.0, 0.0),  
 child: Center(  
 child: ButtonTheme(  
 minWidth: 200.0,  
 height: 30.0,  
 child: RaisedButton(  
 color: Colors.*white*,  
 child: Text(  
 'Check Answer.',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 18.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 onPressed: (){  
 answer = myController.text;  
 isCorrect = (answer == expectedAnswer);  
 accuracyNumber = accuracyDeterminer.nextInt(20);  
 if (accuracyNumber >= 0 && accuracyNumber <= howGood) {  
 setState(() {  
 isSoftwareCorrect = true;  
 });  
 }  
 if (accuracyNumber >= (howGood+1) && accuracyNumber <= 20) {  
 setState(() {  
 isSoftwareCorrect = false;  
 });  
 }  
 answerMessage = (isCorrect == true ) ? "Correct!" : "Sorry, that answer was incorrect.\n The expected answer to this question was $expectedAnswer.";  
 if (isCorrect == true) {  
 setState(() {  
 noCorrect = noCorrect + 1;  
 displayNoCorrect = noCorrect.toString();  
 });  
 }  
 if (isSoftwareCorrect == true) {  
 setState(() {  
 noSoftwareCorrect = noSoftwareCorrect + 1;  
 displaynoSoftwareCorrect = noSoftwareCorrect.toString();  
 });  
 }  
 if (noSoftwareCorrect == noCorrect) {  
 setState(() {  
 winnerStatus = 'Draw';  
 });  
 }  
 if (noSoftwareCorrect > noCorrect) {  
 setState(() {  
 winnerStatus = 'You are losing!';  
 });  
 }  
 if (noSoftwareCorrect < noCorrect) {  
 setState(() {  
 winnerStatus = 'You are winning!';  
 });  
 }  
 Future.delayed(const Duration(milliseconds: 2500), () {  
 setState(() {  
 previous = element;  
 do {  
 element = question.nextInt(noQuestions);  
 } while (previous == element);  
 expectedAnswer = answers.elementAt(element);  
 word = questions.elementAt(element);  
 answerMessage = "Waiting for response....";  
 });  
 });  
 setState(() {  
 counter = counter + 1;  
 displayNumberQ = counter.toString();  
 });  
 },  
 ),  
 ),  
 ),  
 ),  
 Center(  
 child: Padding(  
 padding: const EdgeInsets.fromLTRB(0.0, 10.0, 0.0, 0.0),  
 child: ButtonTheme(  
 minWidth: 200.0,  
 height: 30.0,  
 child: RaisedButton(  
 color: Colors.*white*,  
 child: Text(  
 'Return to Home.',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 18.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 onPressed: (){  
 setState(() {  
 noSoftwareCorrect = 0;  
 noCorrect = 0;  
 counter = 0;  
 answerMessage = "Waiting for response....";  
 winnerStatus = 'Waiting to start battle...';  
 displaynoSoftwareCorrect = noSoftwareCorrect.toString();  
 displayNoCorrect = noCorrect.toString();  
 displayNumberQ = counter.toString();  
 });  
  
 f Navigator.*pushReplacementNamed*(context, '/');  
  
 }  
 ),  
 ),  
 ),  
 ),  
 ],  
 ),  
 ),  
 );  
 }  
}

## ***easytest.dart:***

import 'package:flutter/material.dart';  
import 'package:french\_learning\_ia/pages/score.dart';  
import 'dart:math';  
import 'package:french\_learning\_ia/services/F\_O.dart';  
  
  
  
var noCorrect = 0;  
var counter = 0;  
var NC;  
var expectedAnswerE;  
List<String> questionsE = ['etre','avoir','pouvoir','faire','mettre','dire','devoir','prendre','donner','aller','vouloir','savoir','etre','falloir','lire','finir','manger','prendre','regarder', 'voir', 'venir', 'suivre', 'parler', 'croire', 'aimer', 'passer', 'penser', 'attendre', 'trouver', 'laisser', 'arriver', 'donner', 'appeler', 'partir', 'mettre', 'rester', 'arreter', 'connaitre', 'demander', 'comprendre', 'sortir', 'entendre', 'chercher', 'aider', 'essayer', 'revenir', 'jouer', 'finir', 'perdre', 'sentir',];  
List <String> answersE = ['to be','to have','to be able to','to do','to put','to say','to have to','to take','to give','to go','to want','to know','to be','to have to','to read','to finish','to eat','to take','to look', 'to see', 'to come', 'to follow', 'to talk', 'to believe', 'to like', 'to pass', 'to think', 'to wait for', 'to find', 'to leave', 'to arrive', 'to give', 'to call', 'to go', 'to put', 'to stay', 'to stop', 'to know','to ask', 'to understand', 'to go out', 'to hear', 'to look for', 'to help', 'to try', 'to come back', 'to play', 'to finish', 'to lose', 'to smell',];  
var isCorrect = false;  
var answerMessage = "Waiting for response....";  
var noQuestions = 50;  
var displayNoCorrect;  
var lastScore;  
var myController = TextEditingController();  
Quiz currentTestE = quizGenerator(questionsE,answersE);  
var answerE;  
var wordE;  
class Quiz {  
 List<String> questionList;  
 List<String> answersList;  
 Quiz(List questionsList, List answersList){  
 this.questionList = questionsList;  
 this.answersList = answersList;  
 }  
}  
Quiz quizGenerator (questions, answers) {  
 final Set<int> indexS = Set();  
 for (int i = 0; i <= 150; i++){  
 indexS.add(Random().nextInt(noQuestions-1));  
 }  
 List<String> testQ = [], testA = [];  
 List<int> indexes = indexS.toList();  
 for (int i = 0; i <= 10; i++){  
 var element = indexes[i];  
 testQ.add(questions[element]);  
 testA.add(answers[element]);  
 }  
 var newTest = Quiz(testQ, testA);  
 return newTest;  
}  
class EasyTest extends StatefulWidget {  
 @override  
 \_EasyTestState createState() => \_EasyTestState();  
 void initState () {  
  
 }  
}  
  
class \_EasyTestState extends State<EasyTest> {  
 @override  
 void dispose() {  
 // Clean up the controller when the widget is disposed.  
 super.dispose();  
 }  
 Widget build(BuildContext context) {  
 return Scaffold(  
 backgroundColor: Colors.*orange*[400],  
 appBar: AppBar(  
 title: Text(  
 'Vocabulary Test',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 22.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 leading: IconButton(  
 onPressed: () {  
 Navigator.*pushReplacementNamed*(context, '/');  
 },  
 icon: Icon(Icons.*arrow\_back*),  
 ),  
 backgroundColor: Colors.*deepOrange*,  
 centerTitle: true,  
 ),  
  
 body: SingleChildScrollView(  
 child: Column(  
 children: <Widget>[  
 Padding(  
 padding: const EdgeInsets.fromLTRB(10.0, 10.0, 0.0, 0.0),  
 child: Text(  
 "Question : What is the equivalent of '$wordE' ?",  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 20.0,  
 fontWeight: FontWeight.*bold*,  
 color: Colors.*black*,  
 ),  
 ),  
 ),  
 Padding(  
 padding: const EdgeInsets.fromLTRB(15.0, 30.0, 0.0, 0.0),  
 child: Text(  
 'Answer:',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 20.0,  
 fontWeight: FontWeight.*bold*,  
 color: Colors.*black*,  
 ),  
 ),  
 ),  
 TextField(  
 controller: myController,  
 textAlign: TextAlign.center,  
 decoration: InputDecoration(  
 contentPadding: EdgeInsets.*zero*,  
 hintText: "(only lowercase, no extra spaces)",  
 ),  
 onSubmitted: (answer){  
 },  
 ),  
 SizedBox(height: 50.0),  
 Padding(  
 padding: const EdgeInsets.fromLTRB(0, 0, 0, 0.0),  
 child: Center(  
 child: Text(  
 "Answer Status: $answerMessage",  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 18.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 ),  
 ),  
 Padding(  
 padding: const EdgeInsets.symmetric(vertical: 10.0),  
 child: RaisedButton(  
 color: Colors.*white*,  
 child: Center(  
 child: Text(  
 'Check Answer.',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 18.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 ),  
 onPressed: (){  
 answerE = myController.text;  
 expectedAnswerE = currentTestE.answersList[counter];  
 isCorrect = (answerE == currentTestE.answersList[counter]);  
 answerMessage = (isCorrect == true ) ? "Correct!" : "Sorry, that answer was incorrect.\n The expected answer to this question was $expectedAnswerE.";  
 if (isCorrect == true) {  
 setState(() {  
 noCorrect = noCorrect + 1;  
 });  
 }  
 Future.delayed(const Duration(milliseconds: 2500), () {  
 setState(() {  
 wordE = currentTestE.questionList[counter];  
 answerMessage = "Waiting for response....";  
 });  
 });  
 setState(() {  
 counter = counter + 1;  
 });  
 if (10 == counter) {  
 ScoreOperations.*readScore*().then((score) {  
 setState(() {  
 lastScore = int.*parse*(score);  
 });  
 });  
 Future.delayed(const Duration(milliseconds: 2300), () {  
 ScoreOperations.*saveScore*(noCorrect.toString());  
 });  
 }  
 if (counter == 10){  
 Future.delayed(const Duration(milliseconds: 2650), () {  
 setState(() {  
 NC = noCorrect;  
 displayNoCorrect = noCorrect.toString();  
 counter = 0;  
 noCorrect = 0;  
 });  
 setState(() {  
 currentTestE = quizGenerator(questionsE, answersE);  
 });  
 Navigator.*pushReplacementNamed*(context, '/es');  
 });  
 }  
 },  
 ),  
 ),  
 ],  
 ),  
 ),  
 );  
 }  
}

## ***eta.dart:***

import 'package:flutter/material.dart';  
import 'package:french\_learning\_ia/pages/score.dart';  
import 'dart:math';  
import 'package:french\_learning\_ia/services/F\_O.dart';  
  
  
  
var noCorrect = 0;  
var counter = 0;  
var NC;  
var expectedAnswerEA;  
List <String> answersEA = ['etre','avoir','pouvoir','faire','mettre','dire','devoir','prendre','donner','aller','vouloir','savoir','etre','falloir','lire','finir','manger','prendre','regarder', 'voir', 'venir', 'suivre', 'parler', 'croire', 'aimer', 'passer', 'penser', 'attendre', 'trouver', 'laisser', 'arriver', 'donner', 'appeler', 'partir', 'mettre', 'rester', 'arreter', 'connaitre', 'demander', 'comprendre', 'sortir', 'entendre', 'chercher', 'aider', 'essayer', 'revenir', 'jouer', 'finir', 'perdre', 'sentir',];  
List <String> questionsEA = ['to be','to have','to be able to','to do','to put','to say','to have to','to take','to give','to go','to want','to know','to be','to have to','to read','to finish','to eat','to take','to look', 'to see', 'to come', 'to follow', 'to talk', 'to believe', 'to like', 'to pass', 'to think', 'to wait for', 'to find', 'to leave', 'to arrive', 'to give', 'to call', 'to go', 'to put', 'to stay', ' to stop', 'to know','to ask', 'to understand', 'to go out', 'to hear', 'to look for', 'to help', 'to try', 'to come back', 'to play', 'to finish', 'to lose', 'to smell',];  
var isCorrect = false;  
var answerMessage = "Waiting for response....";  
var noQuestions = 50;  
var displayNoCorrect;  
var lastScore;  
var myController = TextEditingController();  
Quiz currentTestEA = quizGenerator(questionsEA,answersEA);  
var answerEA;  
var wordEA;  
class Quiz {  
 List<String> questionList;  
 List<String> answersList;  
 Quiz(List questionsList, List answersList){  
 this.questionList = questionsList;  
 this.answersList = answersList;  
 }  
}  
Quiz quizGenerator (questions, answers) {  
 final Set<int> indexS = Set();  
 for (int i = 0; i <= 150; i++){  
 indexS.add(Random().nextInt(noQuestions-1));  
 }  
 List<String> testQ = [], testA = [];  
 List<int> indexes = indexS.toList();  
 for (int i = 0; i <= 10; i++){  
 var element = indexes[i];  
 testQ.add(questions[element]);  
 testA.add(answers[element]);  
 }  
 var newTest = Quiz(testQ, testA);  
 return newTest;  
}  
class EasyTestA extends StatefulWidget {  
 @override  
 \_EasyTestAState createState() => \_EasyTestAState();  
 void initState () {  
  
 }  
}  
  
class \_EasyTestAState extends State<EasyTestA> {  
 @override  
 void dispose() {  
 // Clean up the controller when the widget is disposed.  
 super.dispose();  
 }  
 Widget build(BuildContext context) {  
 return Scaffold(  
 backgroundColor: Colors.*orange*[400],  
 appBar: AppBar(  
 title: Text(  
 'Vocabulary Test',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 22.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 leading: IconButton(  
 onPressed: () {  
 Navigator.*pushReplacementNamed*(context, '/');  
 },  
 icon: Icon(Icons.*arrow\_back*),  
 ),  
 backgroundColor: Colors.*deepOrange*,  
 centerTitle: true,  
 ),  
  
 body: SingleChildScrollView(  
 child: Column(  
 children: <Widget>[  
 Padding(  
 padding: const EdgeInsets.fromLTRB(10.0, 10.0, 0.0, 0.0),  
 child: Text(  
 "Question : What is the equivalent of '$wordEA' ?",  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 20.0,  
 fontWeight: FontWeight.*bold*,  
 color: Colors.*black*,  
 ),  
 ),  
 ),  
 Padding(  
 padding: const EdgeInsets.fromLTRB(15.0, 30.0, 0.0, 0.0),  
 child: Text(  
 'Answer:',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 20.0,  
 fontWeight: FontWeight.*bold*,  
 color: Colors.*black*,  
 ),  
 ),  
 ),  
 TextField(  
 controller: myController,  
 textAlign: TextAlign.center,  
 decoration: InputDecoration(  
 contentPadding: EdgeInsets.*zero*,  
 hintText: "(only lowercase, no extra spaces)",  
 ),  
 onSubmitted: (answer){  
 },  
 ),  
 SizedBox(height: 50.0),  
 Padding(  
 padding: const EdgeInsets.fromLTRB(0, 0, 0, 0.0),  
 child: Center(  
 child: Text(  
 "Answer Status: $answerMessage",  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 18.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 ),  
 ),  
 Padding(  
 padding: const EdgeInsets.symmetric(vertical: 10.0),  
 child: RaisedButton(  
 color: Colors.*white*,  
 child: Center(  
 child: Text(  
 'Check Answer.',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 18.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 ),  
 onPressed: (){  
 answerEA = myController.text;  
 expectedAnswerEA = currentTestEA.answersList[counter];  
 isCorrect = (answerEA == currentTestEA.answersList[counter]);  
 answerMessage = (isCorrect == true ) ? "Correct!" : "Sorry, that answer was incorrect.\n The expected answer to this question was $expectedAnswerEA.";  
 if (isCorrect == true) {  
 setState(() {  
 noCorrect = noCorrect + 1;  
 });  
 }  
 Future.delayed(const Duration(milliseconds: 2500), () {  
 setState(() {  
 wordEA = currentTestEA.questionList[counter];  
 answerMessage = "Waiting for response....";  
 });  
 });  
 setState(() {  
 counter = counter + 1;  
 });  
 if (10 == counter) {  
 ScoreOperations.*readScore*().then((score) {  
 setState(() {  
 lastScore = int.*parse*(score);  
 });  
 });  
 Future.delayed(const Duration(milliseconds: 2300), () {  
 ScoreOperations.*saveScore*(noCorrect.toString());  
 });  
 }  
 if (counter == 10){  
 Future.delayed(const Duration(milliseconds: 2650), () {  
 setState(() {  
 NC = noCorrect;  
 displayNoCorrect = noCorrect.toString();  
 counter = 0;  
 noCorrect = 0;  
 });  
 setState(() {  
 currentTestEA = quizGenerator(questionsEA, answersEA);  
 });  
 Navigator.*pushReplacementNamed*(context, '/esa');  
 });  
 }  
 },  
 ),  
 ),  
 ],  
 ),  
 ),  
 );  
 }  
}

## ***test.dart:***

import 'package:flutter/material.dart';  
import 'package:french\_learning\_ia/pages/score.dart';  
import 'dart:math';  
import 'package:french\_learning\_ia/services/F\_O.dart';  
  
  
  
var noCorrect = 0;  
var counter = 0;  
var NC;  
var expectedAnswer;  
var questions = ['rentrer', 'vivre', 'rendre', 'tenir', 'oublier', 'travailler', 'manger', 'entrer', 'devenir', 'commencer', 'payer', 'tirer', 'ouvrir', 'changer', 'excuser', 'dormir', 'occuper', 'marcher', 'envoyer', 'apprendre', 'boire', 'garder', 'montrer', 'asseoir', 'porter', 'prier', 'servir', 'écrire', 'retrouver', 'gagner', 'acheter', 'rappeler', 'lire', 'monter', 'quitter', 'emmener', 'toucher', 'continuer', 'raconter', 'repondre', 'sauver', 'rencontrer', 'fermer', 'valoir', 'compter', 'bouger', 'apporter', 'décider', 'vendre', 'expliquer',];  
var answers = ['to bring in','to live','to return','to hold','to forget','to work','to eat','to go in','to become','to start','to pay','to pull','to open','to change','to forgive','to sleep','to occupy','to walk','to send','to learn','to drink','to look after','to show','to sit down','to carry','to pray','to serve','to write','to find', 'to win', 'to buy', 'to remind', 'to read', 'to go up', 'to leave', 'to take', 'to touch', 'to continue', 'to tell', 'to answer', 'to save', 'to meet', 'to close', 'to hold', 'to count', 'to move', 'to bring', 'to decide', 'to sell', 'to explain',];  
var isCorrect = false;  
var answerMessage = "Waiting for response....";  
var noQuestions = 50;  
var displayNoCorrect;  
var lastScoreM;  
var myController = TextEditingController();  
Quiz currentTest = quizGenerator(questions,answers);  
var answer;  
var word;  
class Quiz {  
 List<String> questionList;  
 List<String> answersList;  
 Quiz(List questionsList, List answersList){  
 this.questionList = questionsList;  
 this.answersList = answersList;  
 }  
}  
Quiz quizGenerator (questions, answers) {  
 final Set<int> indexS = Set();  
 for (int i = 0; i <= 150; i++){  
 indexS.add(Random().nextInt(noQuestions-1));  
 }  
 List<String> testQ = [], testA = [];  
 List<int> indexes = indexS.toList();  
 for (int i = 0; i <= 10; i++){  
 var element = indexes[i];  
 testQ.add(questions[element]);  
 testA.add(answers[element]);  
 }  
 var newTest = Quiz(testQ, testA);  
 return newTest;  
}  
class Test extends StatefulWidget {  
 @override  
 \_TestState createState() => \_TestState();  
 void initState () {  
  
 }  
}  
  
class \_TestState extends State<Test> {  
 @override  
 void dispose() {  
 // Clean up the controller when the widget is disposed.  
 super.dispose();  
 }  
 Widget build(BuildContext context) {  
 return Scaffold(  
 backgroundColor: Colors.*orange*[400],  
 appBar: AppBar(  
 title: Text(  
 'Vocabulary Test',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 22.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 leading: IconButton(  
 onPressed: () {  
 Navigator.*pushReplacementNamed*(context, '/');  
 },  
 icon: Icon(Icons.*arrow\_back*),  
 ),  
 backgroundColor: Colors.*deepOrange*,  
 centerTitle: true,  
 ),  
  
 body: SingleChildScrollView(  
 child: Column(  
 children: <Widget>[  
 Padding(  
 padding: const EdgeInsets.fromLTRB(10.0, 10.0, 0.0, 0.0),  
 child: Text(  
 "Question : What is the equivalent of '$word' ?",  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 20.0,  
 fontWeight: FontWeight.*bold*,  
 color: Colors.*black*,  
 ),  
 ),  
 ),  
 Padding(  
 padding: const EdgeInsets.fromLTRB(15.0, 30.0, 0.0, 0.0),  
 child: Text(  
 'Answer:',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 20.0,  
 fontWeight: FontWeight.*bold*,  
 color: Colors.*black*,  
 ),  
 ),  
 ),  
 TextField(  
 controller: myController,  
 textAlign: TextAlign.center,  
 decoration: InputDecoration(  
 contentPadding: EdgeInsets.*zero*,  
 hintText: "(only lowercase, no extra spaces)",  
 ),  
 onSubmitted: (answer){  
 },  
 ),  
 SizedBox(height: 50.0),  
 Padding(  
 padding: const EdgeInsets.fromLTRB(0, 0, 0, 0.0),  
 child: Center(  
 child: Text(  
 "Answer Status: $answerMessage",  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 18.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 ),  
 ),  
 Padding(  
 padding: const EdgeInsets.symmetric(vertical: 10.0),  
 child: RaisedButton(  
 color: Colors.*white*,  
 child: Center(  
 child: Text(  
 'Check Answer.',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 18.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 ),  
 onPressed: (){  
 answer = myController.text;  
 expectedAnswer = currentTest.answersList[counter];  
 isCorrect = (answer == currentTest.answersList[counter]);  
 answerMessage = (isCorrect == true ) ? "Correct!" : "Sorry, that answer was incorrect.\n The expected answer to this question was $expectedAnswer.";  
 if (isCorrect == true) {  
 setState(() {  
 noCorrect = noCorrect + 1;  
 });  
 }  
 Future.delayed(const Duration(milliseconds: 2500), () {  
 setState(() {  
 word = currentTest.questionList[counter];  
 answerMessage = "Waiting for response....";  
 });  
 });  
 setState(() {  
 counter = counter + 1;  
 });  
 if (10 == counter) {  
 ScoreOperations.*readScore*().then((score) {  
 setState(() {  
 lastScoreM = int.*parse*(score);  
 });  
 });  
 Future.delayed(const Duration(milliseconds: 2300), () {  
 ScoreOperations.*saveScore*(noCorrect.toString());  
 });  
 }  
 if (counter == 10){  
 Future.delayed(const Duration(milliseconds: 2650), () {  
 setState(() {  
 NC = noCorrect;  
 displayNoCorrect = noCorrect.toString();  
 counter = 0;  
 noCorrect = 0;  
 });  
 setState(() {  
 currentTest = quizGenerator(questions, answers);  
 });  
 Navigator.*pushReplacementNamed*(context, '/score');  
 });  
 }  
 },  
 ),  
 ),  
 ],  
 ),  
 ),  
 );  
 }  
}

## ***testA.dart:***

import 'package:flutter/material.dart';  
import 'package:french\_learning\_ia/pages/score.dart';  
import 'dart:math';  
import 'package:french\_learning\_ia/services/F\_O.dart';  
  
  
  
var noCorrect = 0;  
var counter = 0;  
var NC;  
var expectedAnswerA;  
List<String> answersA = ['rentrer', 'vivre', 'rendre', 'tenir', 'oublier', 'travailler', 'manger', 'entrer', 'devenir', 'commencer', 'payer', 'tirer', 'ouvrir', 'changer', 'excuser', 'dormir', 'occuper', 'marcher', 'envoyer', 'apprendre', 'boire', 'garder', 'montrer', 'asseoir', 'porter', 'prier', 'servir', 'écrire', 'retrouver', 'gagner', 'acheter', 'rappeler', 'lire', 'monter', 'quitter', 'emmener', 'toucher', 'continuer', 'raconter', 'repondre', 'sauver', 'rencontrer', 'fermer', 'valoir', 'compter', 'bouger', 'apporter', 'décider', 'vendre', 'expliquer',];  
List<String> questionsA = ['to bring in','to live','to return','to hold','to forget','to work','to eat','to go in','to become','to start','to pay','to pull','to open','to change','to forgive','to sleep','to occupy','to walk','to send','to learn','to drink','to look after','to show','to sit down','to carry','to pray','to serve','to write','to find', 'to win', 'to buy', 'to remind', 'to read', 'to go up', 'to leave', 'to take', 'to touch', 'to continue', 'to tell', 'to answer', 'to save', 'to meet', 'to close', 'to hold', 'to count', 'to move', 'to bring', 'to decide', 'to sell', 'to explain',];  
var isCorrect = false;  
var answerMessage = "Waiting for response....";  
var noQuestions = 50;  
var displayNoCorrect;  
var lastScore;  
var myController = TextEditingController();  
Quiz currentTestA = quizGenerator(questionsA,answersA);  
var answerA;  
var wordA;  
class Quiz {  
 List<String> questionList;  
 List<String> answersList;  
 Quiz(List questionsList, List answersList){  
 this.questionList = questionsList;  
 this.answersList = answersList;  
 }  
}  
Quiz quizGenerator (questions, answers) {  
 final Set<int> indexS = Set();  
 for (int i = 0; i <= 150; i++){  
 indexS.add(Random().nextInt(noQuestions-1));  
 }  
 List<String> testQ = [], testA = [];  
 List<int> indexes = indexS.toList();  
 for (int i = 0; i <= 10; i++){  
 var element = indexes[i];  
 testQ.add(questions[element]);  
 testA.add(answers[element]);  
 }  
 var newTest = Quiz(testQ, testA);  
 return newTest;  
}  
class TestA extends StatefulWidget {  
 @override  
 \_TestAState createState() => \_TestAState();  
 void initState () {  
  
 }  
}  
  
class \_TestAState extends State<TestA> {  
 @override  
 void dispose() {  
 // Clean up the controller when the widget is disposed.  
 super.dispose();  
 }  
 Widget build(BuildContext context) {  
 return Scaffold(  
 backgroundColor: Colors.*orange*[400],  
 appBar: AppBar(  
 title: Text(  
 'Vocabulary Test',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 22.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 leading: IconButton(  
 onPressed: () {  
 Navigator.*pushReplacementNamed*(context, '/');  
 },  
 icon: Icon(Icons.*arrow\_back*),  
 ),  
 backgroundColor: Colors.*deepOrange*,  
 centerTitle: true,  
 ),  
  
 body: SingleChildScrollView(  
 child: Column(  
 children: <Widget>[  
 Padding(  
 padding: const EdgeInsets.fromLTRB(10.0, 10.0, 0.0, 0.0),  
 child: Text(  
 "Question : What is the equivalent of '$wordA' ?",  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 20.0,  
 fontWeight: FontWeight.*bold*,  
 color: Colors.*black*,  
 ),  
 ),  
 ),  
 Padding(  
 padding: const EdgeInsets.fromLTRB(15.0, 30.0, 0.0, 0.0),  
 child: Text(  
 'Answer:',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 20.0,  
 fontWeight: FontWeight.*bold*,  
 color: Colors.*black*,  
 ),  
 ),  
 ),  
 TextField(  
 controller: myController,  
 textAlign: TextAlign.center,  
 decoration: InputDecoration(  
 contentPadding: EdgeInsets.*zero*,  
 hintText: "(only lowercase, no extra spaces)",  
 ),  
 onSubmitted: (answer){  
 },  
 ),  
 SizedBox(height: 50.0),  
 Padding(  
 padding: const EdgeInsets.fromLTRB(0, 0, 0, 0.0),  
 child: Center(  
 child: Text(  
 "Answer Status: $answerMessage",  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 18.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 ),  
 ),  
 Padding(  
 padding: const EdgeInsets.symmetric(vertical: 10.0),  
 child: RaisedButton(  
 color: Colors.*white*,  
 child: Center(  
 child: Text(  
 'Check Answer.',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 18.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 ),  
 onPressed: (){  
 answerA = myController.text;  
 expectedAnswerA = currentTestA.answersList[counter];  
 isCorrect = (answerA == currentTestA.answersList[counter]);  
 answerMessage = (isCorrect == true ) ? "Correct!" : "Sorry, that answer was incorrect.\n The expected answer to this question was $expectedAnswerA.";  
 if (isCorrect == true) {  
 setState(() {  
 noCorrect = noCorrect + 1;  
 });  
 }  
 Future.delayed(const Duration(milliseconds: 2500), () {  
 setState(() {  
 wordA = currentTestA.questionList[counter];  
 answerMessage = "Waiting for response....";  
 });  
 });  
 setState(() {  
 counter = counter + 1;  
 });  
 if (10 == counter) {  
 ScoreOperations.*readScore*().then((score) {  
 setState(() {  
 lastScore = int.*parse*(score);  
 });  
 });  
 Future.delayed(const Duration(milliseconds: 2300), () {  
 ScoreOperations.*saveScore*(noCorrect.toString());  
 });  
 }  
 if (counter == 10){  
 Future.delayed(const Duration(milliseconds: 2650), () {  
 setState(() {  
 NC = noCorrect;  
 displayNoCorrect = noCorrect.toString();  
 counter = 0;  
 noCorrect = 0;  
 });  
 setState(() {  
 currentTestA = quizGenerator(questionsA, answersA);  
 });  
 Navigator.*pushReplacementNamed*(context, '/sa');  
 });  
 }  
 },  
 ),  
 ),  
 ],  
 ),  
 ),  
 );  
 }  
}

### ***hardtest.dart:***

import 'package:flutter/material.dart';  
import 'package:french\_learning\_ia/pages/score.dart';  
import 'dart:math';  
import 'package:french\_learning\_ia/services/F\_O.dart';  
  
  
  
var noCorrect = 0;  
var counter = 0;  
var NC;;  
var expectedAnswerH;  
List <String> questionsH = ['adorer','recevoir','utiliser','coucher','preferer','offrir','preparer','choisir','conduire','chanter','presenter','accepter','refuser', 'terminer','amuser','intéresser','rire','pardonner','embrasser','danser','detester','maintenir','supposer','epouser','approcher','craindre','crier','inviter','arranger','remercier','repeter','signer','accompagner','oser','permettre','annuler','laver','se plaindre','epeler','traduire','eteindre','allumer','se réveiller','ajouter','gouter','couter','cuire','décrire', 'effacer', 'enseigner',];  
List <String> answersH = ['to adore','to receive','to use','to put to bed','to prefer','to offer','to prepare','to choose','to drive','to sing','to introduce','to accept','to refuse', 'to finish','to amuse','to interest','to laugh','to forgive','to kiss','to dance','to hate','to keep','to suppose','to marry','to approach','to fear','to shout','to invite','to arrange','to thank','to repeat','to sign','to go with','to dare','to allow','to cancel','to clean','to complain','to spell', 'to translate', 'to turn off','to turn on','to wake up','to add', 'to taste', 'to cost', 'to cook','to describe', 'to erase', 'to teach',];  
var isCorrect = false;  
var answerMessage = "Waiting for response....";  
var noQuestions = 50;  
var displayNoCorrect;  
var lastScore;  
var myController = TextEditingController();  
Quiz currentTestH = quizGenerator(questionsH,answersH);  
var answerH;  
var wordH;  
class Quiz {  
 List<String> questionList;  
 List<String> answersList;  
 Quiz(List questionsList, List answersList){  
 this.questionList = questionsList;  
 this.answersList = answersList;  
 }  
}  
Quiz quizGenerator (questions, answers) {  
 final Set<int> indexS = Set();  
 for (int i = 0; i <= 150; i++){  
 indexS.add(Random().nextInt(noQuestions-1));  
 }  
 List<String> testQ = [], testA = [];  
 List<int> indexes = indexS.toList();  
 for (int i = 0; i <= 10; i++){  
 var element = indexes[i];  
 testQ.add(questions[element]);  
 testA.add(answers[element]);  
 }  
 var newTest = Quiz(testQ, testA);  
 return newTest;  
}  
class HardTest extends StatefulWidget {  
 @override  
 \_HardTestState createState() => \_HardTestState();  
 void initState () {  
  
 }  
}  
  
class \_HardTestState extends State<HardTest> {  
 @override  
 void dispose() {  
 // Clean up the controller when the widget is disposed.  
 super.dispose();  
 }  
 Widget build(BuildContext context) {  
 return Scaffold(  
 backgroundColor: Colors.*orange*[400],  
 appBar: AppBar(  
 title: Text(  
 'Vocabulary Test',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 22.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 leading: IconButton(  
 onPressed: () {  
 Navigator.*pushReplacementNamed*(context, '/');  
 },  
 icon: Icon(Icons.*arrow\_back*),  
 ),  
 backgroundColor: Colors.*deepOrange*,  
 centerTitle: true,  
 ),  
  
 body: SingleChildScrollView(  
 child: Column(  
 children: <Widget>[  
 Padding(  
 padding: const EdgeInsets.fromLTRB(10.0, 10.0, 0.0, 0.0),  
 child: Text(  
 "Question : What is the equivalent of '$wordH' ?",  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 20.0,  
 fontWeight: FontWeight.*bold*,  
 color: Colors.*black*,  
 ),  
 ),  
 ),  
 Padding(  
 padding: const EdgeInsets.fromLTRB(15.0, 30.0, 0.0, 0.0),  
 child: Text(  
 'Answer:',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 20.0,  
 fontWeight: FontWeight.*bold*,  
 color: Colors.*black*,  
 ),  
 ),  
 ),  
 TextField(  
 controller: myController,  
 textAlign: TextAlign.center,  
 decoration: InputDecoration(  
 contentPadding: EdgeInsets.*zero*,  
 hintText: "(only lowercase, no extra spaces)",  
 ),  
 onSubmitted: (answer){  
 },  
 ),  
 SizedBox(height: 50.0),  
 Padding(  
 padding: const EdgeInsets.fromLTRB(0, 0, 0, 0.0),  
 child: Center(  
 child: Text(  
 "Answer Status: $answerMessage",  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 18.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 ),  
 ),  
 Padding(  
 padding: const EdgeInsets.symmetric(vertical: 10.0),  
 child: RaisedButton(  
 color: Colors.*white*,  
 child: Center(  
 child: Text(  
 'Check Answer.',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 18.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 ),  
 onPressed: (){  
 answerH = myController.text;  
 expectedAnswerH = currentTestH.answersList[counter];  
 isCorrect = (answerH == currentTestH.answersList[counter]);  
 answerMessage = (isCorrect == true ) ? "Correct!" : "Sorry, that answer was incorrect.\n The expected answer to this question was $expectedAnswerH.";  
 if (isCorrect == true) {  
 setState(() {  
 noCorrect = noCorrect + 1;  
 });  
 }  
 Future.delayed(const Duration(milliseconds: 2500), () {  
 setState(() {  
 wordH = currentTestH.questionList[counter];  
 answerMessage = "Waiting for response....";  
 });  
 });  
 setState(() {  
 counter = counter + 1;  
 });  
 if (10 == counter) {  
 ScoreOperations.*readScore*().then((score) {  
 setState(() {  
 lastScore = int.*parse*(score);  
 });  
 });  
 Future.delayed(const Duration(milliseconds: 2300), () {  
 ScoreOperations.*saveScore*(noCorrect.toString());  
 });  
 }  
 if (counter == 10){  
 Future.delayed(const Duration(milliseconds: 2650), () {  
 setState(() {  
 NC = noCorrect;  
 displayNoCorrect = noCorrect.toString();  
 counter = 0;  
 noCorrect = 0;  
 });  
 setState(() {  
 currentTestH = quizGenerator(questionsH, answersH);  
 });  
 Navigator.*pushReplacementNamed*(context, '/hs');  
 });  
 }  
 },  
 ),  
 ),  
 ],  
 ),  
 ),  
 );  
 }  
}

## ***hta.dart:***

import 'package:flutter/material.dart';  
import 'package:french\_learning\_ia/pages/score.dart';  
import 'dart:math';  
import 'package:french\_learning\_ia/services/F\_O.dart';  
  
  
  
var noCorrect = 0;  
var counter = 0;  
var NC;  
var expectedAnswerHA;  
List <String> answersHA = ['adorer','recevoir','utiliser','coucher','preferer','offrir','preparer','choisir','conduire','chanter','presenter','accepter','refuser', 'terminer','amuser','intéresser','rire','pardonner','embrasser','danser','detester','maintenir','supposer','epouser','approcher','craindre','crier','inviter','arranger','remercier','repeter','signer','accompagner','oser','permettre','annuler','laver','se plaindre','epeler','traduire','eteindre','allumer','se réveiller','ajouter','gouter','couter','cuire','décrire', 'effacer', 'enseigner',];  
List <String> questionsHA = ['to adore','to receive','to use','to put to bed','to prefer','to offer','to prepare','to choose','to drive','to sing','to introduce','to accept','to refuse', 'to finish','to amuse','to interest','to laugh','to forgive','to kiss','to dance','to hate','to keep','to suppose','to marry','to approach','to fear','to shout','to invite','to arrange','to thank','to repeat','to sign','to go with','to dare','to allow','to cancel','to clean','to complain','to spell', 'to translate', 'to turn off','to turn on','to wake up','to add', 'to taste', 'to cost', 'to cook','to describe', 'to erase', 'to teach',];  
var isCorrect = false;  
var answerMessage = "Waiting for response....";  
var noQuestions = 50;  
var displayNoCorrect;  
var lastScore;  
var myController = TextEditingController();  
Quiz currentTestHA = quizGenerator(questionsHA,answersHA);  
var answerHA;  
var wordHA;  
class Quiz {  
 List<String> questionList;  
 List<String> answersList;  
 Quiz(List questionsList, List answersList){  
 this.questionList = questionsList;  
 this.answersList = answersList;  
 }  
}  
Quiz quizGenerator (questions, answers) {  
 final Set<int> indexS = Set();  
 for (int i = 0; i <= 150; i++){  
 indexS.add(Random().nextInt(noQuestions-1));  
 }  
 List<String> testQ = [], testA = [];  
 List<int> indexes = indexS.toList();  
 for (int i = 0; i <= 10; i++){  
 var element = indexes[i];  
 testQ.add(questions[element]);  
 testA.add(answers[element]);  
 }  
 var newTest = Quiz(testQ, testA);  
 return newTest;  
}  
class HardTestA extends StatefulWidget {  
 @override  
 \_HardTestAState createState() => \_HardTestAState();  
 void initState () {  
  
 }  
}  
  
class \_HardTestAState extends State<HardTestA> {  
 @override  
 void dispose() {  
 // Clean up the controller when the widget is disposed.  
 super.dispose();  
 }  
 Widget build(BuildContext context) {  
 return Scaffold(  
 backgroundColor: Colors.*orange*[400],  
 appBar: AppBar(  
 title: Text(  
 'Vocabulary Test',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 22.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 leading: IconButton(  
 onPressed: () {  
 Navigator.*pushReplacementNamed*(context, '/');  
 },  
 icon: Icon(Icons.*arrow\_back*),  
 ),  
 backgroundColor: Colors.*deepOrange*,  
 centerTitle: true,  
 ),  
  
 body: SingleChildScrollView(  
 child: Column(  
 children: <Widget>[  
 Padding(  
 padding: const EdgeInsets.fromLTRB(10.0, 10.0, 0.0, 0.0),  
 child: Text(  
 "Question : What is the equivalent of '$wordHA' ?",  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 20.0,  
 fontWeight: FontWeight.*bold*,  
 color: Colors.*black*,  
 ),  
 ),  
 ),  
 Padding(  
 padding: const EdgeInsets.fromLTRB(15.0, 30.0, 0.0, 0.0),  
 child: Text(  
 'Answer:',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 20.0,  
 fontWeight: FontWeight.*bold*,  
 color: Colors.*black*,  
 ),  
 ),  
 ),  
 TextField(  
 controller: myController,  
 textAlign: TextAlign.center,  
 decoration: InputDecoration(  
 contentPadding: EdgeInsets.*zero*,  
 hintText: "(only lowercase, no extra spaces)",  
 ),  
 onSubmitted: (answer){  
 },  
 ),  
 SizedBox(height: 50.0),  
 Padding(  
 padding: const EdgeInsets.fromLTRB(0, 0, 0, 0.0),  
 child: Center(  
 child: Text(  
 "Answer Status: $answerMessage",  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 18.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 ),  
 ),  
 Padding(  
 padding: const EdgeInsets.symmetric(vertical: 10.0),  
 child: RaisedButton(  
 color: Colors.*white*,  
 child: Center(  
 child: Text(  
 'Check Answer.',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 18.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 ),  
 onPressed: (){  
 answerHA = myController.text;  
 expectedAnswerHA = currentTestHA.answersList[counter];  
 isCorrect = (answerHA == currentTestHA.answersList[counter]);  
 answerMessage = (isCorrect == true ) ? "Correct!" : "Sorry, that answer was incorrect.\n The expected answer to this question was $expectedAnswerHA.";  
 if (isCorrect == true) {  
 setState(() {  
 noCorrect = noCorrect + 1;  
 });  
 }  
 Future.delayed(const Duration(milliseconds: 2500), () {  
 setState(() {  
 wordHA = currentTestHA.questionList[counter];  
 answerMessage = "Waiting for response....";  
 });  
 });  
 setState(() {  
 counter = counter + 1;  
 });  
 if (10 == counter) {  
 ScoreOperations.*readScore*().then((score) {  
 setState(() {  
 lastScore = int.*parse*(score);  
 });  
 });  
 Future.delayed(const Duration(milliseconds: 2300), () {  
 ScoreOperations.*saveScore*(noCorrect.toString());  
 });  
 }  
 if (counter == 10){  
 Future.delayed(const Duration(milliseconds: 2650), () {  
 setState(() {  
 NC = noCorrect;  
 displayNoCorrect = noCorrect.toString();  
 counter = 0;  
 noCorrect = 0;  
 });  
 setState(() {  
 currentTestHA = quizGenerator(questionsHA, answersHA);  
 });  
 Navigator.*pushReplacementNamed*(context, '/hsa');  
 });  
 }  
 },  
 ),  
 ),  
 ],  
 ),  
 ),  
 );  
 }  
}

|  |  |
| --- | --- |
|  |  |

### ***score.dart:***

import 'package:flutter/material.dart';  
import 'package:french\_learning\_ia/pages/test.dart';  
  
var currentScore;  
var scoreChange;  
var displayscoreChange = (NC - lastScoreM).toString();  
var displayProgressText;  
var recommendation;  
  
class Score extends StatefulWidget {  
 @override  
 \_ScoreState createState() => \_ScoreState();  
}  
  
class \_ScoreState extends State<Score> {  
 @override  
 Widget build(BuildContext context) {  
 return Scaffold(  
 backgroundColor: Colors.*orange*[400],  
 appBar: AppBar(  
 title: Text(  
 'Test Results',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 22.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 backgroundColor: Colors.*deepOrange*,  
 centerTitle: true,  
 ),  
 body: Center(  
 child: Column(  
 children: <Widget>[  
 Padding(  
 padding: const EdgeInsets.fromLTRB(15.0, 0.0, 0.0, 0.0),  
 child: Text(  
 'You got $displayNoCorrect/10.',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 22.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 ),  
 Padding(  
 padding: const EdgeInsets.fromLTRB(15.0, 0.0, 0.0, 0.0),  
 child: Text(  
 'Would you like to take another test?',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 22.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 ),  
 Padding(  
 padding: const EdgeInsets.fromLTRB(0.0,10,0,10.0),  
 child: ButtonTheme(  
 minWidth: 200.0,  
 height: 30.0,  
 child: RaisedButton(  
 color: Colors.*white*,  
 child: Text(  
 'View Progress',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 18.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 onPressed: () {  
 setState(() {  
 scoreChange = NC -lastScoreM;  
 displayscoreChange = scoreChange.toString();  
 });  
 if (scoreChange > 0) {  
 setState(() {  
 displayProgressText = "You've improved, keep it up!";  
 });  
 }  
 if (scoreChange < 0) {  
 setState(() {  
 displayProgressText = "Your score has decreased since your last test.";  
 });  
 }  
 if (scoreChange == 0) {  
 setState(() {  
 displayProgressText = "You're score hasn't changed.";  
 });  
 }  
 if (NC == lastScoreM) {  
 setState(() {  
 displayscoreChange = '0';  
 });  
 }  
 if (NC >= 0 && NC < 5) {  
 setState(() {  
 recommendation = "Easy";  
 });  
 }  
 if (NC >= 5 && NC < 8) {  
 setState(() {  
 recommendation = "Medium";  
 });  
 }  
 if (NC >= 8 && NC <= 10) {  
 setState(() {  
 recommendation = "Hard";  
 });  
 }  
 print (displayProgressText);  
 Navigator.*pushReplacementNamed*(context, '/progress');  
 },  
 ),  
 ),  
 ),  
 Padding(  
 padding: const EdgeInsets.fromLTRB(0.0,10,0,10.0),  
 child: ButtonTheme(  
 minWidth: 200.0,  
 height: 30.0,  
 child: RaisedButton(  
 color: Colors.*white*,  
 child: Text(  
 'Return to Home.',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 18.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 onPressed: (){  
 Navigator.*pushReplacementNamed*(context, '/');  
 },  
 ),  
 ),  
 ),  
 Padding(  
 padding: const EdgeInsets.fromLTRB(0.0,10,0,10.0),  
 child: ButtonTheme(  
 minWidth: 200.0,  
 height: 30.0,  
 child: RaisedButton(  
 color: Colors.*white*,  
 child: Text(  
 'Take Another Test',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 18.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 onPressed: (){  
 Navigator.*pushReplacementNamed*(context, '/test');  
 },  
 ),  
 ),  
 ),  
 ],  
 ),  
 )  
 );  
 }  
}

|  |  |
| --- | --- |
|  |  |

## ***scoreE.dart:***

import 'package:flutter/material.dart';  
import 'package:french\_learning\_ia/pages/easytest.dart';  
  
var currentScore;  
var scoreChange;  
var displayscoreChange = (NC - lastScore).toString();  
var displayProgressText;  
var recommendation;  
  
class ScoreE extends StatefulWidget {  
 @override  
 \_ScoreEState createState() => \_ScoreEState();  
}  
  
class \_ScoreEState extends State<ScoreE> {  
 @override  
 Widget build(BuildContext context) {  
 return Scaffold(  
 backgroundColor: Colors.*orange*[400],  
 appBar: AppBar(  
 title: Text(  
 'Test Results',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 22.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 backgroundColor: Colors.*deepOrange*,  
 centerTitle: true,  
 ),  
 body: Center(  
 child: Column(  
 children: <Widget>[  
 Padding(  
 padding: const EdgeInsets.fromLTRB(15.0, 10.0, 0.0, 10.0),  
 child: Text(  
 'You got $displayNoCorrect/10.',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 22.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 ),  
 Padding(  
 padding: const EdgeInsets.fromLTRB(15.0, 0.0, 0.0, 10.0),  
 child: Text(  
 'Would you like to take another test?',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 22.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 ),  
 Padding(  
 padding: const EdgeInsets.fromLTRB(0.0,0,0,0),  
 child: ButtonTheme(  
 minWidth: 200.0,  
 height: 30.0,  
 child: RaisedButton(  
 color: Colors.*white*,  
 child: Text(  
 'View Progress',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 18.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 onPressed: () {  
 setState(() {  
 scoreChange = NC -lastScore;  
 displayscoreChange = scoreChange.toString();  
 });  
 if (scoreChange > 0) {  
 setState(() {  
 displayProgressText = "You've improved, keep it up!";  
 });  
 }  
 if (scoreChange < 0) {  
 setState(() {  
 displayProgressText = "Your score has decreased since your last test.";  
 });  
 }  
 if (scoreChange == 0) {  
 setState(() {  
 displayProgressText = "You're score hasn't changed.";  
 });  
 }  
 if (NC == lastScore) {  
 setState(() {  
 displayscoreChange = '0';  
 });  
 }  
 if (NC >= 0 && NC < 5) {  
 setState(() {  
 recommendation = "Easy";  
 });  
 }  
 if (NC >= 5 && NC < 8) {  
 setState(() {  
 recommendation = "Easy";  
 });  
 }  
 if (NC >= 8 && NC <= 10) {  
 setState(() {  
 recommendation = "Medium";  
 });  
 }  
 print (displayProgressText);  
 Navigator.*pushReplacementNamed*(context, '/ep');  
 },  
 ),  
 ),  
 ),  
 Padding(  
 padding: const EdgeInsets.fromLTRB(0.0,10,0,10.0),  
 child: ButtonTheme(  
 minWidth: 200.0,  
 height: 30.0,  
 child: RaisedButton(  
 color: Colors.*white*,  
 child: Text(  
 'Return to Home.',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 18.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 onPressed: (){  
 Navigator.*pushReplacementNamed*(context, '/');  
 },  
 ),  
 ),  
 ),  
 Padding(  
 padding: const EdgeInsets.fromLTRB(0.0,0,0,10.0),  
 child: ButtonTheme(  
 minWidth: 200.0,  
 height: 30.0,  
 child: RaisedButton(  
 color: Colors.*white*,  
 child: Text(  
 'Take Another Test',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 18.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 onPressed: (){  
 Navigator.*pushReplacementNamed*(context, '/et');  
 },  
 ),  
 ),  
 ),  
 ],  
 ),  
 )  
 );  
 }  
}

|  |  |
| --- | --- |
|  |  |

### ***scoreH.dart:***

import 'package:flutter/material.dart';  
import 'package:french\_learning\_ia/pages/hardtest.dart';  
  
var currentScoreH;  
var scoreChangeH;  
var displayscoreChangeH = (NC - lastScore).toString();  
var displayProgressTextH;  
var recommendationH;  
  
class ScoreH extends StatefulWidget {  
 @override  
 \_ScoreHState createState() => \_ScoreHState();  
}  
  
class \_ScoreHState extends State<ScoreH> {  
 @override  
 Widget build(BuildContext context) {  
 return Scaffold(  
 backgroundColor: Colors.*orange*[400],  
 appBar: AppBar(  
 title: Text(  
 'Test Results',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 22.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 backgroundColor: Colors.*deepOrange*,  
 centerTitle: true,  
 ),  
 body: Center(  
 child: Column(  
 children: <Widget>[  
 Padding(  
 padding: const EdgeInsets.fromLTRB(15.0, 0.0, 0.0, 0.0),  
 child: Text(  
 'You got $displayNoCorrect/10.',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 22.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 ),  
 Padding(  
 padding: const EdgeInsets.fromLTRB(15.0, 0.0, 0.0, 0.0),  
 child: Text(  
 'Would you like to take another test?',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 22.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 ),  
 Padding(  
 padding: const EdgeInsets.fromLTRB(0.0,10,0,10.0),  
 child: ButtonTheme(  
 minWidth: 200.0,  
 height: 30.0,  
 child: RaisedButton(  
 color: Colors.*white*,  
 child: Text(  
 'View Progress',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 18.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 onPressed: () {  
 setState(() {  
 scoreChangeH = NC -lastScore;  
 displayscoreChangeH = scoreChangeH.toString();  
 });  
 if (scoreChangeH > 0) {  
 setState(() {  
 displayProgressTextH = "You've improved, keep it up!";  
 });  
 }  
 if (scoreChangeH < 0) {  
 setState(() {  
 displayProgressTextH = "Your score has decreased since your last test.";  
 });  
 }  
 if (scoreChangeH == 0) {  
 setState(() {  
 displayProgressTextH = "You're score hasn't changed.";  
 });  
 }  
 if (NC == lastScore) {  
 setState(() {  
 displayscoreChangeH = '0';  
 });  
 }  
 if (NC >= 0 && NC < 5) {  
 setState(() {  
 recommendationH = "Medium";  
 });  
 }  
 if (NC >= 5 && NC < 8) {  
 setState(() {  
 recommendationH = "Hard";  
 });  
 }  
 if (NC >= 8 && NC <= 10) {  
 setState(() {  
 recommendationH = "Hard";  
 });  
 }  
 print (displayProgressTextH);  
 Navigator.*pushReplacementNamed*(context, '/hp');  
 },  
 ),  
 ),  
 ),  
 Padding(  
 padding: const EdgeInsets.fromLTRB(0.0,10,0,10.0),  
 child: ButtonTheme(  
 minWidth: 200.0,  
 height: 30.0,  
 child: RaisedButton(  
 color: Colors.*white*,  
 child: Text(  
 'Return to Home.',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 18.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 onPressed: (){  
 Navigator.*pushReplacementNamed*(context, '/');  
 },  
 ),  
 ),  
 ),  
 Padding(  
 padding: const EdgeInsets.fromLTRB(0.0,10,0,10.0),  
 child: ButtonTheme(  
 minWidth: 200.0,  
 height: 30.0,  
 child: RaisedButton(  
 color: Colors.*white*,  
 child: Text(  
 'Take Another Test',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 18.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 onPressed: (){  
 Navigator.*pushReplacementNamed*(context, '/ht');  
 },  
 ),  
 ),  
 ),  
 ],  
 ),  
 )  
 );  
 }  
}

## ***sa.dart:***

import 'package:flutter/material.dart';  
import 'package:french\_learning\_ia/pages/testA.dart';  
  
var currentScoreA;  
var scoreChangeA;  
var displayscoreChangeA = (NC - lastScore).toString();  
var displayProgressTextA;  
var recommendationA;  
  
class ScoreA extends StatefulWidget {  
 @override  
 \_ScoreAState createState() => \_ScoreAState();  
}  
  
class \_ScoreAState extends State<ScoreA> {  
 @override  
 Widget build(BuildContext context) {  
 return Scaffold(  
 backgroundColor: Colors.*orange*[400],  
 appBar: AppBar(  
 title: Text(  
 'Test Results',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 22.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 backgroundColor: Colors.*deepOrange*,  
 centerTitle: true,  
 ),  
 body: Center(  
 child: Column(  
 children: <Widget>[  
 Padding(  
 padding: const EdgeInsets.fromLTRB(15.0, 0.0, 0.0, 0.0),  
 child: Text(  
 'You got $displayNoCorrect/10.',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 22.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 ),  
 Padding(  
 padding: const EdgeInsets.fromLTRB(15.0, 0.0, 0.0, 0.0),  
 child: Text(  
 'Would you like to take another test?',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 22.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 ),  
 Padding(  
 padding: const EdgeInsets.fromLTRB(0.0,0,0,10.0),  
 child: ButtonTheme(  
 minWidth: 200.0,  
 height: 30.0,  
 child: RaisedButton(  
 color: Colors.*white*,  
 child: Text(  
 'View Progress',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 18.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 onPressed: () {  
 setState(() {  
 scoreChangeA = NC -lastScore;  
 displayscoreChangeA = scoreChangeA.toString();  
 });  
 if (scoreChangeA> 0) {  
 setState(() {  
 displayProgressTextA = "You've improved, keep it up!";  
 });  
 }  
 if (scoreChangeA < 0) {  
 setState(() {  
 displayProgressTextA = "Your score has decreased since your last test.";  
 });  
 }  
 if (scoreChangeA == 0) {  
 setState(() {  
 displayProgressTextA = "You're score hasn't changed.";  
 });  
 }  
 if (NC == lastScore) {  
 setState(() {  
 displayscoreChangeA = '0';  
 });  
 }  
 if (NC >= 0 && NC < 5) {  
 setState(() {  
 recommendationA = "Easy";  
 });  
 }  
 if (NC >= 5 && NC < 8) {  
 setState(() {  
 recommendationA = "Medium";  
 });  
 }  
 if (NC >= 8 && NC <= 10) {  
 setState(() {  
 recommendationA = "Hard";  
 });  
 }  
 print (displayProgressTextA);  
 Navigator.*pushReplacementNamed*(context, '/pa');  
 },  
 ),  
 ),  
 ),  
 Padding(  
 padding: const EdgeInsets.fromLTRB(0.0,0,0,10.0),  
 child: ButtonTheme(  
 minWidth: 200.0,  
 height: 30.0,  
 child: RaisedButton(  
 color: Colors.*white*,  
 child: Text(  
 'Return to Home.',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 18.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 onPressed: (){  
 Navigator.*pushReplacementNamed*(context, '/');  
 },  
 ),  
 ),  
 ),  
 Padding(  
 padding: const EdgeInsets.fromLTRB(0.0,0,0,10.0),  
 child: ButtonTheme(  
 minWidth: 200.0,  
 height: 30.0,  
 child: RaisedButton(  
 color: Colors.*white*,  
 child: Text(  
 'Take Another Test',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 18.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 onPressed: (){  
 Navigator.*pushReplacementNamed*(context, '/testA');  
 },  
 ),  
 ),  
 ),  
 ],  
 ),  
 )  
 );  
 }  
}

## ***esa.dart:***

import 'package:flutter/material.dart';  
import 'package:french\_learning\_ia/pages/eta.dart';  
  
var currentScoreEA;  
var scoreChangeEA;  
var displayscoreChangeEA = (NC - lastScore).toString();  
var displayProgressTextEA;  
var recommendationEA;  
  
class ScoreEA extends StatefulWidget {  
 @override  
 \_ScoreEAState createState() => \_ScoreEAState();  
}  
  
class \_ScoreEAState extends State<ScoreEA> {  
 @override  
 Widget build(BuildContext context) {  
 return Scaffold(  
 backgroundColor: Colors.*orange*[400],  
 appBar: AppBar(  
 title: Text(  
 'Test Results',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 22.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 backgroundColor: Colors.*deepOrange*,  
 centerTitle: true,  
 ),  
 body: Center(  
 child: Column(  
 children: <Widget>[  
 Padding(  
 padding: const EdgeInsets.fromLTRB(15.0, 0.0, 0.0, 0.0),  
 child: Text(  
 'You got $displayNoCorrect/10.',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 22.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 ),  
 Padding(  
 padding: const EdgeInsets.fromLTRB(15.0, 0.0, 0.0, 0.0),  
 child: Text(  
 'Would you like to take another test?',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 22.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 ),  
 Padding(  
 padding: const EdgeInsets.fromLTRB(0.0,10,0,10.0),  
 child: ButtonTheme(  
 minWidth: 200.0,  
 height: 30.0,  
 child: RaisedButton(  
 color: Colors.*white*,  
 child: Text(  
 'View Progress',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 18.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 onPressed: () {  
 setState(() {  
 scoreChangeEA = NC -lastScore;  
 displayscoreChangeEA = scoreChangeEA.toString();  
 });  
 if (scoreChangeEA > 0) {  
 setState(() {  
 displayProgressTextEA = "You've improved, keep it up!";  
 });  
 }  
 if (scoreChangeEA < 0) {  
 setState(() {  
 displayProgressTextEA = "Your score has decreased since your last test.";  
 });  
 }  
 if (scoreChangeEA == 0) {  
 setState(() {  
 displayProgressTextEA = "You're score hasn't changed.";  
 });  
 }  
 if (NC == lastScore) {  
 setState(() {  
 displayscoreChangeEA = '0';  
 });  
 }  
 if (NC >= 0 && NC < 5) {  
 setState(() {  
 recommendationEA = "Easy";  
 });  
 }  
 if (NC >= 5 && NC < 8) {  
 setState(() {  
 recommendationEA = "Easy";  
 });  
 }  
 if (NC >= 8 && NC <= 10) {  
 setState(() {  
 recommendationEA = "Medium";  
 });  
 }  
 print (displayProgressTextEA);  
 Navigator.*pushReplacementNamed*(context, '/epa');  
 },  
 ),  
 ),  
 ),  
 Padding(  
 padding: const EdgeInsets.fromLTRB(0.0,0,0,10.0),  
 child: ButtonTheme(  
 minWidth: 200.0,  
 height: 30.0,  
 child: RaisedButton(  
 color: Colors.*white*,  
 child: Text(  
 'Return to Home.',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 18.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 onPressed: (){  
 Navigator.*pushReplacementNamed*(context, '/');  
 },  
 ),  
 ),  
 ),  
 Padding(  
 padding: const EdgeInsets.fromLTRB(0.0,0,0,10.0),  
 child: ButtonTheme(  
 minWidth: 200.0,  
 height: 30.0,  
 child: RaisedButton(  
 color: Colors.*white*,  
 child: Text(  
 'Take Another Test',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 18.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 onPressed: (){  
 Navigator.*pushReplacementNamed*(context, '/eta');  
 },  
 ),  
 ),  
 ),  
 ],  
 ),  
 )  
 );  
 }  
}

## ***hsa.dart:***

import 'package:flutter/material.dart';  
import 'package:french\_learning\_ia/pages/hta.dart';  
  
var currentScore;  
var scoreChange;  
var displayscoreChange = (NC - lastScore).toString();  
var displayProgressText;  
var recommendation;  
  
class ScoreHA extends StatefulWidget {  
 @override  
 \_ScoreHAState createState() => \_ScoreHAState();  
}  
  
class \_ScoreHAState extends State<ScoreHA> {  
 @override  
 Widget build(BuildContext context) {  
 return Scaffold(  
 backgroundColor: Colors.*orange*[400],  
 appBar: AppBar(  
 title: Text(  
 'Test Results',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 22.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 backgroundColor: Colors.*deepOrange*,  
 centerTitle: true,  
 ),  
 body: Center(  
 child: Column(  
 children: <Widget>[  
 Padding(  
 padding: const EdgeInsets.fromLTRB(15.0, 0.0, 0.0, 0.0),  
 child: Text(  
 'You got $displayNoCorrect/10.',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 22.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 ),  
 Padding(  
 padding: const EdgeInsets.fromLTRB(15.0, 0.0, 0.0, 0.0),  
 child: Text(  
 'Would you like to take another test?',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 22.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 ),  
 ButtonTheme(  
 minWidth: 200.0,  
 height: 30.0,  
 child: Padding(  
 padding: const EdgeInsets.fromLTRB(0.0,0,0,10.0),  
 child: RaisedButton(  
 color: Colors.*white*,  
 child: Text(  
 'View Progress',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 18.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 onPressed: () {  
 setState(() {  
 scoreChange = NC -lastScore;  
 displayscoreChange = scoreChange.toString();  
 });  
 if (scoreChange > 0) {  
 setState(() {  
 displayProgressText = "You've improved, keep it up!";  
 });  
 }  
 if (scoreChange < 0) {  
 setState(() {  
 displayProgressText = "Your score has decreased since your last test.";  
 });  
 }  
 if (scoreChange == 0) {  
 setState(() {  
 displayProgressText = "You're score hasn't changed.";  
 });  
 }  
 if (NC == lastScore) {  
 setState(() {  
 displayscoreChange = '0';  
 });  
 }  
 if (NC >= 0 && NC < 5) {  
 setState(() {  
 recommendation = "Medium";  
 });  
 }  
 if (NC >= 5 && NC < 8) {  
 setState(() {  
 recommendation = "Hard";  
 });  
 }  
 if (NC >= 8 && NC <= 10) {  
 setState(() {  
 recommendation = "Hard";  
 });  
 }  
 print (displayProgressText);  
 Navigator.*pushReplacementNamed*(context, '/hpa');  
 },  
 ),  
 ),  
 ),  
 Padding(  
 padding: const EdgeInsets.fromLTRB(0.0,0,0,10.0),  
 child: ButtonTheme(  
 minWidth: 200.0,  
 height: 30.0,  
 child: RaisedButton(  
 color: Colors.*white*,  
 child: Text(  
 'Return to Home.',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 18.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 onPressed: (){  
 Navigator.*pushReplacementNamed*(context, '/');  
 },  
 ),  
 ),  
 ),  
 Padding(  
 padding: const EdgeInsets.fromLTRB(0.0,0,0,10.0),  
 child: ButtonTheme(  
 minWidth: 200.0,  
 height: 30.0,  
 child: RaisedButton(  
 color: Colors.*white*,  
 child: Text(  
 'Take Another Test',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 18.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 onPressed: (){  
 Navigator.*pushReplacementNamed*(context, '/hta');  
 },  
 ),  
 ),  
 ),  
 ],  
 ),  
 )  
 );  
 }  
}

## ***progress.dart:***

import 'package:flutter/material.dart';  
import 'package:french\_learning\_ia/pages/score.dart';  
  
class Progress extends StatefulWidget {  
 @override  
 \_ProgressState createState() => \_ProgressState();  
}  
  
class \_ProgressState extends State<Progress> {  
 @override  
 Widget build(BuildContext context) {  
 return Scaffold(  
 backgroundColor: Colors.*orange*[400],  
 appBar: AppBar(  
 title: Text(  
 'Progress Screen',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 22.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 backgroundColor: Colors.*deepOrange*,  
 centerTitle: true,  
 ),  
 body: Center(  
 child: Column(  
 children: [  
 Padding(  
 padding: const EdgeInsets.all(8.0),  
 child: Center(  
 child: Text(  
 'Change of marks since last test: $displayscoreChange',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 18.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 ),  
 ),  
 Padding(  
 padding: const EdgeInsets.all(8.0),  
 child: Center(  
 child: Text(  
 '$displayProgressText',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 18.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 ),  
 ),  
 Padding(  
 padding: const EdgeInsets.all(8.0),  
 child: Center(  
 child: Text(  
 'Difficulty Level Recommendation: $recommendation',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 18.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 ),  
 ),  
 Padding(  
 padding: const EdgeInsets.symmetric(vertical: 10.0),  
 child: RaisedButton(  
 color: Colors.*white*,  
 child: Center(  
 child: Text(  
 'Return to Home.',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 18.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 ),  
 onPressed: (){  
 Navigator.*pushReplacementNamed*(context, '/');  
 },  
 ),  
 ),  
 ],  
 ),  
 ),  
 );  
 }  
}

|  |  |
| --- | --- |
|  |  |

## ***progressE.dart:***

import 'package:flutter/material.dart';  
import 'package:french\_learning\_ia/pages/scoreE.dart';  
  
class ProgressE extends StatefulWidget {  
 @override  
 \_ProgressEState createState() => \_ProgressEState();  
}  
  
const sep = 20.0;  
class \_ProgressEState extends State<ProgressE> {  
 @override  
 Widget build(BuildContext context) {  
 return Scaffold(  
 backgroundColor: Colors.*orange*[400],  
 appBar: AppBar(  
 title: Text(  
 'Progress Screen',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 22.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 backgroundColor: Colors.*deepOrange*,  
 centerTitle: true,  
 ),  
 body: Center(  
 child: Column(  
 children: [  
 Padding(  
 padding: const EdgeInsets.fromLTRB(0.0,sep,0.0, 0.0),  
 child: Center(  
 child: Text(  
 'Change of marks since last test: $displayscoreChange',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 18.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 ),  
 ),  
 Padding(  
 padding: const EdgeInsets.fromLTRB(0.0, sep, 0.0, 0.0),  
 child: Center(  
 child: Text(  
 '$displayProgressText',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 18.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 ),  
 ),  
 Padding(  
 padding: const EdgeInsets.fromLTRB(0.0, sep, 0.0, 0.0),  
 child: Center(  
 child: Text(  
 'Difficulty Level Recommendation: $recommendation',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 18.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 ),  
 ),  
 Padding(  
 padding: const EdgeInsets.symmetric(vertical: 10.0),  
 child: RaisedButton(  
 color: Colors.*white*,  
 child: Center(  
 child: Text(  
 'Return to Home.',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 18.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 ),  
 onPressed: (){  
 Navigator.*pushReplacementNamed*(context, '/');  
 },  
 ),  
 ),  
 ],  
 ),  
 ),  
 );  
 }  
}

# ***progressH.dart:***

import 'package:flutter/material.dart';  
import 'package:french\_learning\_ia/pages/scoreH.dart';  
  
class ProgressH extends StatefulWidget {  
 @override  
 \_ProgressHState createState() => \_ProgressHState();  
}  
  
class \_ProgressHState extends State<ProgressH> {  
 @override  
 Widget build(BuildContext context) {  
 return Scaffold(  
 backgroundColor: Colors.*orange*[400],  
 appBar: AppBar(  
 title: Text(  
 'Progress Screen',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 22.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 backgroundColor: Colors.*deepOrange*,  
 centerTitle: true,  
 ),  
 body: Center(  
 child: Column(  
 children: [  
 Padding(  
 padding: const EdgeInsets.all(8.0),  
 child: Center(  
 child: Text(  
 'Change of marks since last test: $displayscoreChangeH',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 18.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 ),  
 ),  
 Padding(  
 padding: const EdgeInsets.all(8.0),  
 child: Center(  
 child: Text(  
 '$displayProgressTextH',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 18.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 ),  
 ),  
 Padding(  
 padding: const EdgeInsets.all(8.0),  
 child: Center(  
 child: Text(  
 'Difficulty Level Recommendation: $recommendationH',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 18.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 ),  
 ),  
 Padding(  
 padding: const EdgeInsets.symmetric(vertical: 10.0),  
 child: RaisedButton(  
 color: Colors.*white*,  
 child: Center(  
 child: Text(  
 'Return to Home.',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 18.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 ),  
 onPressed: (){  
 Navigator.*pushReplacementNamed*(context, '/');  
 },  
 ),  
 )  
 ],  
 ),  
 ),  
 );  
 }  
}

***pa.dart:***

import 'package:flutter/material.dart';  
import 'package:french\_learning\_ia/pages/sa.dart';  
  
class ProgressA extends StatefulWidget {  
 @override  
 \_ProgressAState createState() => \_ProgressAState();  
}  
  
class \_ProgressAState extends State<ProgressA> {  
 @override  
 Widget build(BuildContext context) {  
 return Scaffold(  
 backgroundColor: Colors.*orange*[400],  
 appBar: AppBar(  
 title: Text(  
 'Progress Screen',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 22.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 backgroundColor: Colors.*deepOrange*,  
 centerTitle: true,  
 ),  
 body: Center(  
 child: Column(  
 children: [  
 Padding(  
 padding: const EdgeInsets.all(8.0),  
 child: Center(  
 child: Text(  
 'Change of marks since last test: $displayscoreChangeA',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 18.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 ),  
 ),  
 Padding(  
 padding: const EdgeInsets.all(8.0),  
 child: Center(  
 child: Text(  
 '$displayProgressTextA',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 18.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 ),  
 ),  
 Padding(  
 padding: const EdgeInsets.all(8.0),  
 child: Center(  
 child: Text(  
 'Difficulty Level Recommendation: $recommendationA',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 18.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 ),  
 ),  
 Padding(  
 padding: const EdgeInsets.symmetric(vertical: 10.0),  
 child: RaisedButton(  
 color: Colors.*white*,  
 child: Center(  
 child: Text(  
 'Return to Home.',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 18.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 ),  
 onPressed: (){  
 Navigator.*pushReplacementNamed*(context, '/');  
 },  
 ),  
 ),  
 ],  
 ),  
 ),  
 );  
 }  
}

# ***pea.dart:***

import 'package:flutter/material.dart';  
import 'package:french\_learning\_ia/pages/esa.dart';  
  
class ProgressEA extends StatefulWidget {  
 @override  
 \_ProgressEAState createState() => \_ProgressEAState();  
}  
  
class \_ProgressEAState extends State<ProgressEA> {  
 @override  
 Widget build(BuildContext context) {  
 return Scaffold(  
 backgroundColor: Colors.*orange*[400],  
 appBar: AppBar(  
 title: Text(  
 'Progress Screen',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 22.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 backgroundColor: Colors.*deepOrange*,  
 centerTitle: true,  
 ),  
 body: Center(  
 child: Column(  
 children: [  
 Padding(  
 padding: const EdgeInsets.all(8.0),  
 child: Center(  
 child: Text(  
 'Change of marks since last test : $displayscoreChangeEA',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 18.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 ),  
 ),  
 Padding(  
 padding: const EdgeInsets.all(8.0),  
 child: Center(  
 child: Text(  
 '$displayProgressTextEA',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 18.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 ),  
 ),  
 Padding(  
 padding: const EdgeInsets.all(8.0),  
 child: Center(  
 child: Text(  
 'Difficulty Level Recommendation: $recommendationEA',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 18.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 ),  
 ),  
 Padding(  
 padding: const EdgeInsets.symmetric(vertical: 10.0),  
 child: RaisedButton(  
 color: Colors.*white*,  
 child: Center(  
 child: Text(  
 'Return to Home.',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 18.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 ),  
 onPressed: (){  
 Navigator.*pushReplacementNamed*(context, '/');  
 },  
 ),  
 ),  
 ],  
 ),  
 ),  
 );  
 }  
}

# ***hpa.dart:***

import 'package:flutter/material.dart';  
import 'package:french\_learning\_ia/pages/hsa.dart';  
  
class ProgressHA extends StatefulWidget {  
 @override  
 \_ProgressHAState createState() => \_ProgressHAState();  
}  
  
class \_ProgressHAState extends State<ProgressHA> {  
 @override  
 Widget build(BuildContext context) {  
 return Scaffold(  
 backgroundColor: Colors.*orange*[400],  
 appBar: AppBar(  
 title: Text(  
 'Progress Screen',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 22.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 backgroundColor: Colors.*deepOrange*,  
 centerTitle: true,  
 ),  
 body: Center(  
 child: Column(  
 children: [  
 Padding(  
 padding: const EdgeInsets.all(8.0),  
 child: Center(  
 child: Text(  
 'Change of marks since last test: $displayscoreChange',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 18.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 ),  
 ),  
 Padding(  
 padding: const EdgeInsets.all(8.0),  
 child: Center(  
 child: Text(  
 'Difficulty Level Recommendation: $recommendation',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 18.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 ),  
 ),  
 Padding(  
 padding: const EdgeInsets.all(8.0),  
 child: Center(  
 child: Text(  
 '$displayProgressText',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 18.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 ),  
 ),  
 Padding(  
 padding: const EdgeInsets.symmetric(vertical: 10.0),  
 child: RaisedButton(  
 color: Colors.*white*,  
 child: Center(  
 child: Text(  
 'Return to Home.',  
 style: TextStyle(  
 fontFamily: 'Oswald',  
 fontSize: 18.0,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 ),  
 onPressed: (){  
 Navigator.*pushReplacementNamed*(context, '/');  
 },  
 ),  
 )  
  
 ],  
 ),  
 ),  
 );  
 }  
}

# **Services:**

***F\_O.dart:***

import 'package:path\_provider/path\_provider.dart';  
import 'dart:io';  
  
class ScoreOperations {  
 static Future<String> get *getScorePath* async {  
 final scorePath = await getApplicationDocumentsDirectory();  
 return scorePath.path;  
 }  
  
 static Future<File> get *getScore* async {  
 final scoreP = await *getScorePath*;  
 return File('$scoreP/lastScore.txt');  
 }  
  
 static Future<File> *saveScore*(String data) async {  
 final scoreFile = await *getScore*;  
 return scoreFile.writeAsString(data);  
 }  
  
 static Future<String> *readScore*() async {  
 try {  
 final scoreFile = await *getScore*;  
 String pastScore = await scoreFile.readAsString();  
 return pastScore;  
 } catch (e) {  
 return "";  
 }  
 }  
}

# **Other:**

***main.dart:***

import 'package:flutter/material.dart';  
import 'package:french\_learning\_ia/pages/battlemode.dart';  
import 'package:french\_learning\_ia/pages/bmeasy.dart';  
import 'package:french\_learning\_ia/pages/bmhard.dart';  
import 'package:french\_learning\_ia/pages/bmmedium.dart';  
import 'package:french\_learning\_ia/pages/easytest.dart';  
import 'package:french\_learning\_ia/pages/esa.dart';  
import 'package:french\_learning\_ia/pages/eta.dart';  
import 'package:french\_learning\_ia/pages/hardtest.dart';  
import 'package:french\_learning\_ia/pages/home.dart';  
import 'package:french\_learning\_ia/pages/hpa.dart';  
import 'package:french\_learning\_ia/pages/hsa.dart';  
import 'package:french\_learning\_ia/pages/hta.dart';  
import 'package:french\_learning\_ia/pages/pa.dart';  
import 'package:french\_learning\_ia/pages/pea.dart';  
import 'package:french\_learning\_ia/pages/progress.dart';  
import 'package:french\_learning\_ia/pages/sa.dart';  
import 'package:french\_learning\_ia/pages/scoreE.dart';  
import 'package:french\_learning\_ia/pages/scoreH.dart';  
import 'package:french\_learning\_ia/pages/test.dart';  
import 'package:french\_learning\_ia/pages/score.dart';  
import 'package:french\_learning\_ia/pages/progressE.dart';  
import 'package:french\_learning\_ia/pages/progressH.dart';  
import 'package:french\_learning\_ia/pages/testA.dart';  
  
void main() => runApp(MaterialApp(  
 initialRoute: '/',  
 routes: {  
 '/': (context) => Home(),  
 '/test': (context) => Test(),  
 '/testA': (context) => TestA(),  
 '/score': (context) => Score(),  
 '/progress': (context) => Progress(),  
 '/ht': (context) => HardTest(),  
 '/hta': (context) => HardTestA(),  
 '/et': (context) => EasyTest(),  
 '/eta': (context) => EasyTestA(),  
 '/es': (context) => ScoreE(),  
 '/hs': (context) => ScoreH(),  
 '/ep': (context) => ProgressE(),  
 '/hp': (context) => ProgressH(),  
 '/esa': (context) => ScoreEA(),  
 '/epa': (context) => ProgressEA(),  
 '/hpa': (context) => ProgressHA(),  
 '/ta': (context) => TestA(),  
 '/hsa': (context) => ScoreHA(),  
 '/sa': (context) => ScoreA(),  
 '/pa': (context) => ProgressA(),  
 '/bm': (context) => BattleMode(),  
 '/bme': (context) => EasyBM(),  
 '/bmm': (context) => MediumBM(),  
 '/bmh': (context) => HardBM(),  
 }  
));

## ***pubspec.yaml:***

name: french\_learning\_ia  
description: A new Flutter application.  
  
*# The following line prevents the package from being accidentally published to  
#* [*pub.dev*](http://pub.dev/) *using `pub publish`. This is preferred for private packages.*publish\_to: 'none' *# Remove this line if you wish to publish to* [*pub.dev*](http://pub.dev/) *# The following defines the version and build number for your application.  
# A version number is three numbers separated by dots, like 1.2.43  
# followed by an optional build number separated by a +.  
# Both the version and the builder number may be overridden in flutter  
# build by specifying --build-name and --build-number, respectively.  
# In Android, build-name is used as versionName while build-number used as versionCode.  
# Read more about Android versioning at* [*https://developer.android.com/studio/publish/versioning*](https://developer.android.com/studio/publish/versioning) *# In iOS, build-name is used as CFBundleShortVersionString while build-number used as CFBundleVersion.  
# Read more about iOS versioning at  
#* [*https://developer.apple.com/library/archive/documentation/General/Reference/InfoPlistKeyReference/Articles/CoreFoundationKeys.html*](https://developer.apple.com/library/archive/documentation/General/Reference/InfoPlistKeyReference/Articles/CoreFoundationKeys.html)version: 1.0.0+1  
  
environment:  
 sdk: ">=2.7.0 <3.0.0"  
  
dependencies:  
 flutter:  
 sdk: flutter  
  
  
 *# The following adds the Cupertino Icons font to your application.  
 # Use with the CupertinoIcons class for iOS style icons.* cupertino\_icons: ^0.1.3  
 path\_provider: ^1.6.24  
  
dev\_dependencies:  
 flutter\_test:  
 sdk: flutter  
  
*# For information on the generic Dart part of this file, see the  
# following page:* [*https://dart.dev/tools/pub/pubspec*](https://dart.dev/tools/pub/pubspec) *# The following section is specific to Flutter.*flutter:  
  
 *# The following line ensures that the Material Icons font is  
 # included with your application, so that you can use the icons in  
 # the material Icons class.* uses-material-design: true  
  
 *# To add assets to your application, add an assets section, like this:  
 # assets:  
 # - images/a\_dot\_burr.jpeg  
 # - images/a\_dot\_ham.jpeg  
  
 # An image asset can refer to one or more resolution-specific "variants", see  
 #* [*https://flutter.dev/assets-and-images/#resolution-aware*](https://flutter.dev/assets-and-images/#resolution-aware)*.  
  
 # For details regarding adding assets from package dependencies, see  
 #* [*https://flutter.dev/assets-and-images/#from-packages*](https://flutter.dev/assets-and-images/#from-packages) *# To add custom fonts to your application, add a fonts section here,  
 # in this "flutter" section. Each entry in this list should have a  
 # "family" key with the font family name, and a "fonts" key with a  
 # list giving the asset and other descriptors for the font. For  
 # example:* fonts:  
 - family: Oswald  
 fonts:  
 - asset: fonts/Oswald-Regular.ttf