import 'package:flutter/material.dart';

import 'package:flutter/services.dart';

void main() {

runApp(const MyApp());

}

class MyApp extends StatelessWidget {

const MyApp({super.key});

// This widget is the root of your application.

@override

Widget build(BuildContext context) {

return MaterialApp(

title: 'Flutter Demo',

theme: ThemeData(

// This is the theme of your application.

//

// Try running your application with "flutter run". You'll see the

// application has a blue toolbar. Then, without quitting the app, try

// changing the primarySwatch below to Colors.green and then invoke

// "hot reload" (press "r" in the console where you ran "flutter run",

// or simply save your changes to "hot reload" in a Flutter IDE).

// Notice that the counter didn't reset back to zero; the application

// is not restarted.

primarySwatch: Colors.blue,

),

debugShowCheckedModeBanner: false,

home: const MyHomePage(title: 'Squash Scoring Software – Have Fun),

);

}

}

class MyHomePage extends StatefulWidget {

const MyHomePage({super.key, required this.title});

// This widget is the home page of your application. It is stateful, meaning

// that it has a State object (defined below) that contains fields that affect

// how it looks.

// This class is the configuration for the state. It holds the values (in this

// case the title) provided by the parent (in this case the App widget) and

// used by the build method of the State. Fields in a Widget subclass are

// always marked "final".

final String title;

@override

State<MyHomePage> createState() => \_MyHomePageState();

}

class \_MyHomePageState extends State<MyHomePage> {

int \_counter1 = 0;

int \_counter2 = 0;

int \_setNo = 1;

int \_limit\_c1 = 10;

int \_limit\_c2 = 10;

var \_controller1 = TextEditingController();

var \_controller2 = TextEditingController();

int \_S1P1 = 0;

int \_S1P2 = 0;

int \_S2P1 = 0;

int \_S2P2 = 0;

int \_S3P1 = 0;

int \_S3P2 = 0;

String \_ResultR1 = "";

String \_ResultR2 = "";

bool setResetFlag = false;

double f1 = 300;

final fc = FocusNode();

void \_clearControllers()

{

\_controller1.clear();

\_controller2.clear();

}

void \_incrementCounter1() {

setState(() {

// This call to setState tells the Flutter framework that something has

// changed in this State, which causes it to rerun the build method below

// so that the display can reflect the updated values. If we changed

// \_counter without calling setState(), then the build method would not be

// called again, and so nothing would appear to happen.

\_counter1++;

if (\_counter1==9 && \_counter2==8 )

{

\_limit\_c1 = 20;

\_limit\_c2 = 20;

f1 = 180;

}

if(\_counter1==\_limit\_c1)

{

\_counter1=9;

}

});

}

void \_decCounter1() {

setState(() {

// This call to setState tells the Flutter framework that something has

// changed in this State, which causes it to rerun the build method below

// so that the display can reflect the updated values. If we changed

// \_counter without calling setState(), then the build method would not be

// called again, and so nothing would appear to happen.

\_counter1--;

if(\_counter1==-1)

{

\_counter1=0;

}

});

}

void \_decCounter2() {

setState(() {

// This call to setState tells the Flutter framework that something has

// changed in this State, which causes it to rerun the build method below

// so that the display can reflect the updated values. If we changed

// \_counter without calling setState(), then the build method would not be

// called again, and so nothing would appear to happen.

\_counter2--;

if(\_counter2==-1)

{

\_counter2=0;

}

});

}

void \_incrementCounter2() {

setState(() {

// This call to setState tells the Flutter framework that something has

// changed in this State, which causes it to rerun the build method below

// so that the display can reflect the updated values. If we changed

// \_counter without calling setState(), then the build method would not be

// called again, and so nothing would appear to happen.

\_counter2++;

if (\_counter1==8 && \_counter2==9 )

{

\_limit\_c1 = 20;

\_limit\_c2 = 20;

f1 = 150;

}

if(\_counter2==\_limit\_c2)

{

\_counter2=9;

}

});

}

void \_resetCounters() {

setState(() {

// This call to setState tells the Flutter framework that something has

// changed in this State, which causes it to rerun the build method below

// so that the display can reflect the updated values. If we changed

// \_counter without calling setState(), then the build method would not be

// called again, and so nothing would appear to happen.

\_counter1 = 0;

\_counter2 = 0;

\_S1P1 = 0;

\_S1P2 = 0;

\_S2P1 = 0;

\_S2P2 = 0;

\_S3P1 = 0;

\_S3P2 = 0;

\_ResultR1 = "";

\_ResultR2 = "";

});

}

void \_incrementSetNo() {

setState(() {

// This call to setState tells the Flutter framework that something has

// changed in this State, which causes it to rerun the build method below

// so that the display can reflect the updated values. If we changed

// \_counter without calling setState(), then the build method would not be

// called again, and so nothing would appear to happen.

\_setNo ++;

if(\_setNo==4 && setResetFlag == true)

{

\_setNo=1;

setResetFlag = false;

\_resetCounters();

}

if (\_setNo == 2)

{

if (setResetFlag == false){

\_S1P1 = \_counter1;

\_S1P2 = \_counter2;

}

}

if (\_setNo == 3)

{

if(setResetFlag == false)

{

\_S2P1 = \_counter1;

\_S2P2 = \_counter2;

}

}

if (\_setNo == 4) {

if (setResetFlag == false) {

\_S3P1 = \_counter1;

\_S3P2 = \_counter2;

}

if (\_S1P1 > \_S1P2 && \_S2P1 > \_S2P2) {

\_ResultR1 = "Winner";

\_ResultR2 = "";

}

else if (\_S2P1 > \_S2P2 && \_S3P1 > \_S3P2)

{

\_ResultR1 = "Winner";

\_ResultR2 = "";

}

else if ( \_S1P1 > \_S1P2 && \_S2P2 > \_S2P1 && \_S3P1 > \_S3P2)

{

{

\_ResultR1 = "Winner";

\_ResultR2 = "";

}

}

else if (\_S1P1 == 0 && \_S2P1 == 0 && \_S3P1 == 0 && \_S1P2 == 0 && \_S2P2 == 0 && \_S3P2 == 0 )

{

\_ResultR1 = "";

\_ResultR2 = "";

}

else {

\_ResultR2 = "Winner";

\_ResultR1 = "";

}

\_setNo = 3;

setResetFlag = true;

}

\_limit\_c1 = 10;

\_limit\_c2 = 10;

\_counter1 = 0;

\_counter2 = 0;

f1 = 300;

});

}

@override

Widget build(BuildContext context) {

// This method is rerun every time setState is called, for instance as done

// by the \_incrementCounter method above.

//

// The Flutter framework has been optimized to make rerunning build methods

// fast, so that you can just rebuild anything that needs updating rather

// than having to individually change instances of widgets.

return Scaffold(

appBar: null,

/\*appBar: AppBar(

title: Text("SCORE MANAGEMENT APPLICATION"),

centerTitle: true,

),\*/

body: RawKeyboardListener(

focusNode: fc,

autofocus: true,

onKey: (event) {

if (event.isKeyPressed(LogicalKeyboardKey.arrowUp)) { // for Player 2 inc

// if (event is RawKeyDownEvent) {

\_incrementCounter2();

print("I am pressed");

// }

}

if (event.isKeyPressed(LogicalKeyboardKey.arrowDown )) { // for Player 2 dec

\_decCounter2();

}

if (event.isKeyPressed(LogicalKeyboardKey.keyW)) { // for Player 2 dec

\_incrementCounter1();

}

if (event.isKeyPressed(LogicalKeyboardKey.keyS)) { // for Player 2 dec

\_decCounter1();

}

if (event.isKeyPressed(LogicalKeyboardKey.keyL)) { // for Player 2 dec

\_incrementSetNo();

}

},

child: Center(

// Center is a layout widget. It takes a single child and positions it

// in the middle of the parent.

child: Container(

decoration: BoxDecoration(

image: DecorationImage(

image: AssetImage("assets/images/backgr.png"),

fit: BoxFit.cover,

),

),

// Column is also a layout widget. It takes a list of children and

// arranges them vertically. By default, it sizes itself to fit its

// children horizontally, and tries to be as tall as its parent.

//

// Invoke "debug painting" (press "p" in the console, choose the

// "Toggle Debug Paint" action from the Flutter Inspector in Android

// Studio, or the "Toggle Debug Paint" command in Visual Studio Code)

// to see the wireframe for each widget.

//

// Column has various properties to control how it sizes itself and

// how it positions its children. Here we use mainAxisAlignment to

// center the children vertically; the main axis here is the vertical

// axis because Columns are vertical (the cross axis would be

// horizontal).

child: Column(

crossAxisAlignment: CrossAxisAlignment.center,

children: [

//child: Text("Behzad"),

Row(

mainAxisAlignment: MainAxisAlignment.spaceEvenly,

children: [

Expanded(child: Image.asset('assets/images/div\_s.png')),

Text("HAVE FUN - SQUASH CHAMPIONSHIP", style: TextStyle(fontWeight: FontWeight.bold, fontSize: 60,decoration: TextDecoration.underline,), ),

Expanded(child: Image.asset('assets/images/div2\_s.png')),

],

),

Row(

mainAxisAlignment: MainAxisAlignment.spaceEvenly,

children: [

Expanded(child: TextField(

controller: \_controller1,

textAlign: TextAlign.center,

decoration: InputDecoration(

border: OutlineInputBorder(),

hintText: '1st Player Name',

),

style: TextStyle(fontSize: 60, fontWeight: FontWeight.bold),

),),

Expanded(

child: TextField(

controller: \_controller2,

textAlign: TextAlign.center,

decoration: InputDecoration(

border: OutlineInputBorder(),

hintText: '2nd Player Name',

),

style: TextStyle(fontSize: 60, fontWeight: FontWeight.bold),

),),

],

),

Row(

mainAxisAlignment: MainAxisAlignment.end,

children: [

TextButton(

style: ButtonStyle(

foregroundColor: MaterialStateProperty.all<Color>(Colors.red),

),

onPressed: \_clearControllers,

child: Text('Reset Names'),

)

],

),

/\* Row(

mainAxisAlignment: MainAxisAlignment.spaceEvenly,

children: [

new InkWell(

onTap: \_incrementSetNo,

child: Text(

textAlign: TextAlign.center,

"Set No."+"$\_setNo",

style: TextStyle(fontSize: 50, fontWeight: FontWeight.bold),

),

),

],

), \*/

Row(

mainAxisAlignment: MainAxisAlignment.spaceEvenly,

children: [

SizedBox(

width: 230,

child: new GestureDetector(

onTap: \_incrementCounter1,

child: Text(

textAlign: TextAlign.center,

'$\_counter1',

style: TextStyle(fontSize: f1, fontWeight: FontWeight.bold),

),

),),

/\* new InkWell(

onTap: \_incrementCounter1,

child: Text(

textAlign: TextAlign.center,

'$\_counter1',

style: TextStyle(fontSize: 230, fontWeight: FontWeight.bold),

),

), \*/

// Expanded(child: Text(

// textAlign: TextAlign.center,

// '$\_counter1',

// style: TextStyle(fontSize: 250, fontWeight: FontWeight.bold),

// ),

// ),

//SizedBox(width: 50),

Column(

mainAxisAlignment: MainAxisAlignment.spaceEvenly,

children: [

SizedBox(

height: 200,

width: 230,

child: new GestureDetector(

onTap: \_incrementSetNo,

child: Text(

textAlign: TextAlign.center,

"Set No."+"$\_setNo",

style: TextStyle(fontSize: 50, fontWeight: FontWeight.bold),

),

),),

//SizedBox(height: 20),

Text('Score Board', style: TextStyle(fontSize: 40, fontWeight: FontWeight.bold, decoration: TextDecoration.underline),),

Table(

//defaultColumnWidth: FixedColumnWidth(80.0),

columnWidths: {

0: FixedColumnWidth(190),

1: FixedColumnWidth(80),

2: FixedColumnWidth(80),

3: FixedColumnWidth(80),

4: FixedColumnWidth(190),

},

border: TableBorder.all(

color: Colors.black,

style: BorderStyle.solid,

width: 2),

children: [

TableRow( children: [

Column(children:[Text('Player', style: TextStyle(fontSize: 25.0, fontWeight: FontWeight.bold))]),

Column(children:[Text('Set 1', style: TextStyle(fontSize: 25.0, fontWeight: FontWeight.bold))]),

Column(children:[Text('Set 2', style: TextStyle(fontSize: 25.0, fontWeight: FontWeight.bold))]),

Column(children:[Text('Set 3', style: TextStyle(fontSize: 25.0, fontWeight: FontWeight.bold))]),

Column(children:[Text(' Final Result', style: TextStyle(fontSize: 25.0, fontWeight: FontWeight.bold))]),

]),

TableRow(

children: [

Column(children:[Text(\_controller1.text,style: TextStyle( fontSize: 20.0,fontWeight: FontWeight.bold)), ]),

Column(children:[Text("$\_S1P1", style: TextStyle( fontSize: 20.0,fontWeight: FontWeight.bold))]),

Column(children:[Text("$\_S2P1", style: TextStyle( fontSize: 20.0,fontWeight: FontWeight.bold))]),

Column(children:[Text("$\_S3P1", style: TextStyle( fontSize: 20.0,fontWeight: FontWeight.bold))]),

Column(children:[Text("$\_ResultR1", style: TextStyle( fontSize: 20.0,fontWeight: FontWeight.bold))]),

]),

TableRow( children: [

Column(children:[Text(\_controller2.text, style: TextStyle( fontSize: 20.0,fontWeight: FontWeight.bold))]),

Column(children:[Text("$\_S1P2", style: TextStyle( fontSize: 20.0,fontWeight: FontWeight.bold))]),

Column(children:[Text("$\_S2P2", style: TextStyle( fontSize: 20.0,fontWeight: FontWeight.bold))]),

Column(children:[Text("$\_S3P2",style: TextStyle( fontSize: 20.0,fontWeight: FontWeight.bold))]),

Column(children:[Text("$\_ResultR2", style: TextStyle( fontSize: 20.0,fontWeight: FontWeight.bold))]),

]),

],

),

],

),

SizedBox(

width: 230,

child : new GestureDetector(

onTap: \_incrementCounter2,

child: Text(

textAlign: TextAlign.center,

'$\_counter2',

style: TextStyle(fontSize: f1, fontWeight: FontWeight.bold),

),

),

),

/\* new GestureDetector(

onTap: \_incrementCounter2,

child: Text(

textAlign: TextAlign.center,

'$\_counter2',

style: TextStyle(fontSize: 300, fontWeight: FontWeight.bold),

),

), \*/

],

),

Row(

mainAxisAlignment: MainAxisAlignment.spaceEvenly,

children: [

Expanded(child: TextButton(

style: ButtonStyle(

foregroundColor: MaterialStateProperty.all<Color>(Colors.red),

),

onPressed: \_decCounter1,

child: Text('Dec Point'),

)

),

SizedBox(width: 600),

Expanded(child: TextButton(

style: ButtonStyle(

foregroundColor: MaterialStateProperty.all<Color>(Colors.red),

),

onPressed: \_decCounter2,

child: Text('Dec Point'),

)

),

],

),

//const Text('You have pushed the button this many times:',),

//Text(

// '$\_counter1',

// style: Theme.of(context).textTheme.headline4,

//),

],

),

/\* const Text('You have pushed the button this many times:',),

Text(

'$\_counter',

style: Theme.of(context).textTheme.headline4,

),

\*/

),

),

),

floatingActionButton: FloatingActionButton(

onPressed: \_resetCounters,

tooltip: 'Increment',

child: const Icon(Icons.lock\_reset),

), // This trailing comma makes auto-formatting nicer for build methods.

);

}

}