How to Build a Flutter™ Card List In Less Than 10 Minutes

DLT Labs · Dec 9, 2019★

In our FlutterIn5Series, we build widgets in 5 simple steps. This is the first lesson, where we are going to build a basic card layout to represent cryptocurrency data as shown in the image.

Don’t know Flutter? Don’t worry!

If you are in the field of mobile development, you might already know about flutter. Those who do not have any clue about mobile development, don’t worry. We recently explored flutter to help people understand what it is about.

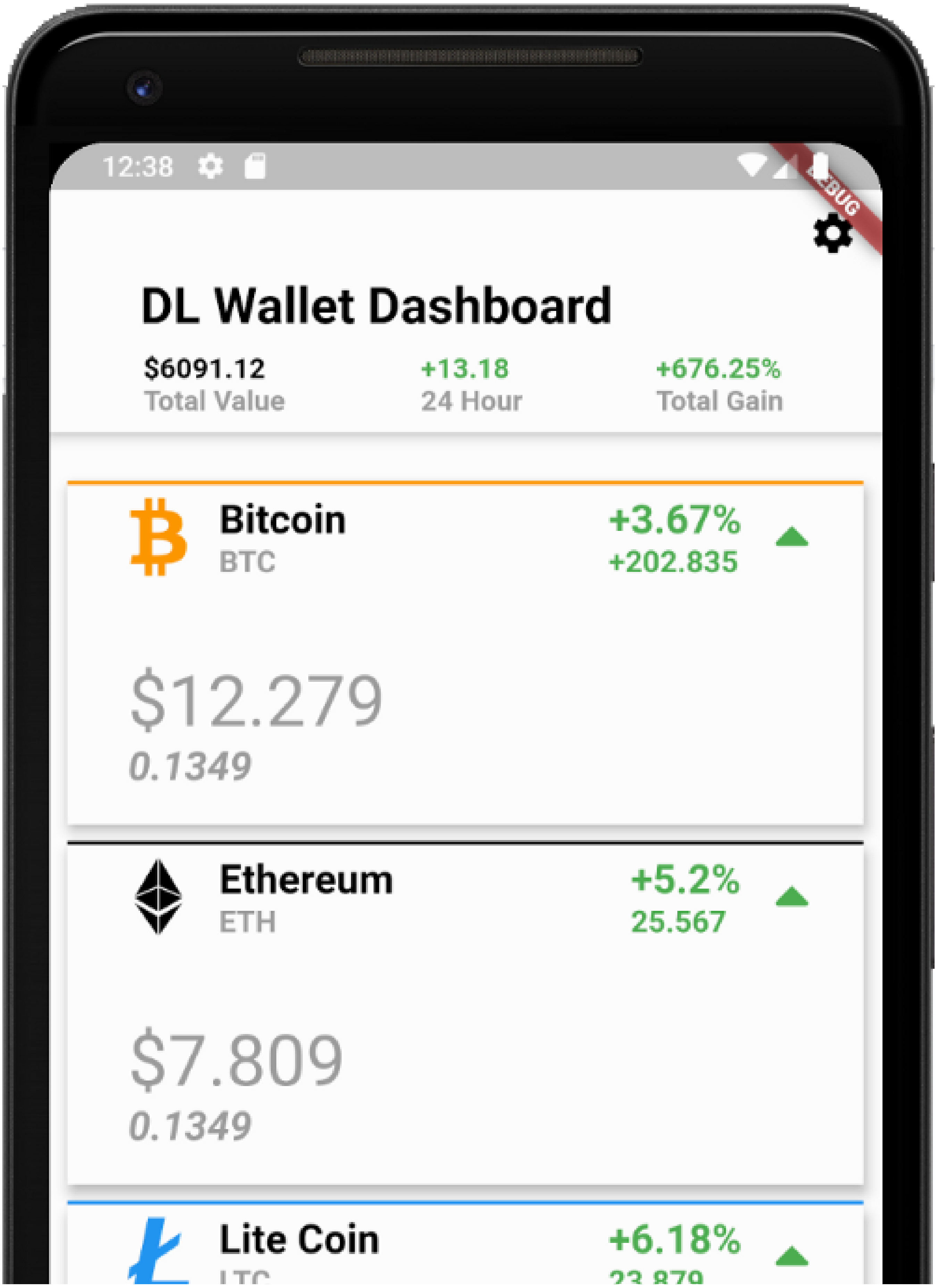
Flutter — A cross development platform for mobile apps

Flutter is becoming a serious competitor of other mobile development platforms like a native, hybrid or other…

DLT Labs on medium.com

The main advantage of flutter is that you can directly dive into the development of flutter without any prior knowledge on Android and iOS development. Sounds cool, right? Then let's go ahead without further ado!

What do you need?



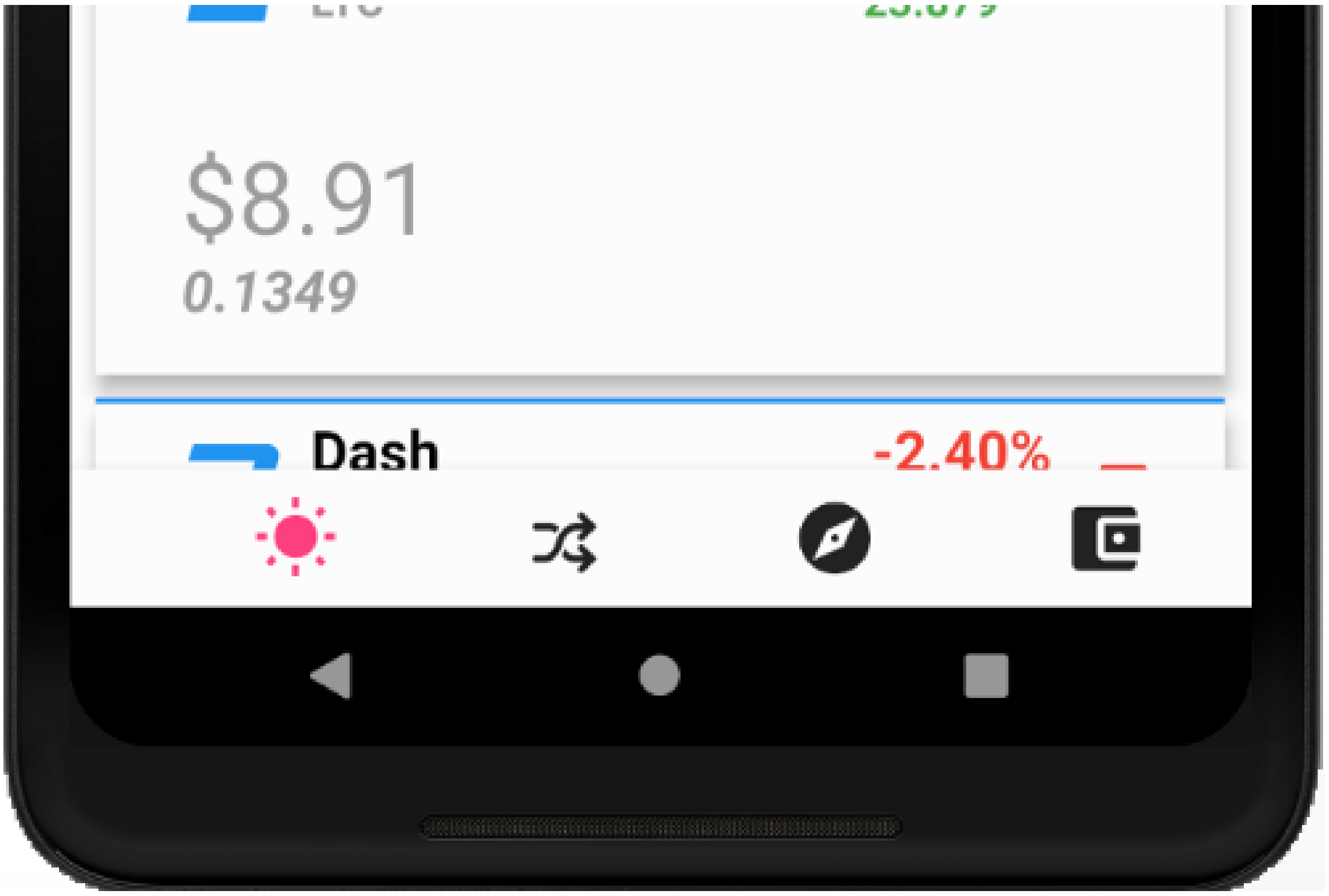
DL Wallet Dashboard

$6091.12	+13.18	+676.25%
Total Value	24 Hour	Total Gain

Bitcoin
BTC
+3.67%
+202.835
$12.279
0.1349

Ethereum
ETH
+5.2%
25.567
$7.809
0.1349

Lite Coin
LTC
+6.18%
23.879



You need to have **flutter sdk** in your system, which you will find at this link:

> *https://flutter.dev/docs/get-started/install* .

Additionally, you need these two packages to use Icons:

crypto_font_icons: ^0.1.0+1 typicons_flutter: ^0.2.1

Add these in your **pubspec.yaml** and run the get command (it will be prompted in IntelliJ and run automatically in VS Code respectively).

Do I need prior knowledge of mobile development?

Absolutely not. I myself didn't know any of the basics of mobile development. All you need is some interest and a little intelligence to leverage your prior coding knowledge to learn and create with flutter.

. . .

The 5 steps to building a flutter card list:

Step 1 - Basic Project Creation and setup

Once you create the basic project (you can use this link for reference: https://flutter.dev/docs/get-started/test-drive?tab=vscode), create a file called **dashboard.dart** under *lib folder*.

Create a stateless widget (preferably using Flutter Live Templates) with the name '**Dashboard**'. Return a MaterialApp and include a scaffold widget as its child and a **Hello World** text to check the rendering.

Now clear all the code in main.dart, except the *runApp* command. Replace the runApp command line with this code:

```
void main() => runApp(Dashboard());
```

> *This will open the* ***Dashboard*** *widget when the app runs.*

The code in dashboard.dart looks like this:

```
import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';
class Dashboard extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
   return MaterialApp(
     home: Scaffold(
       body: Text('Hello World'),
     ),
   );
 }
}
```

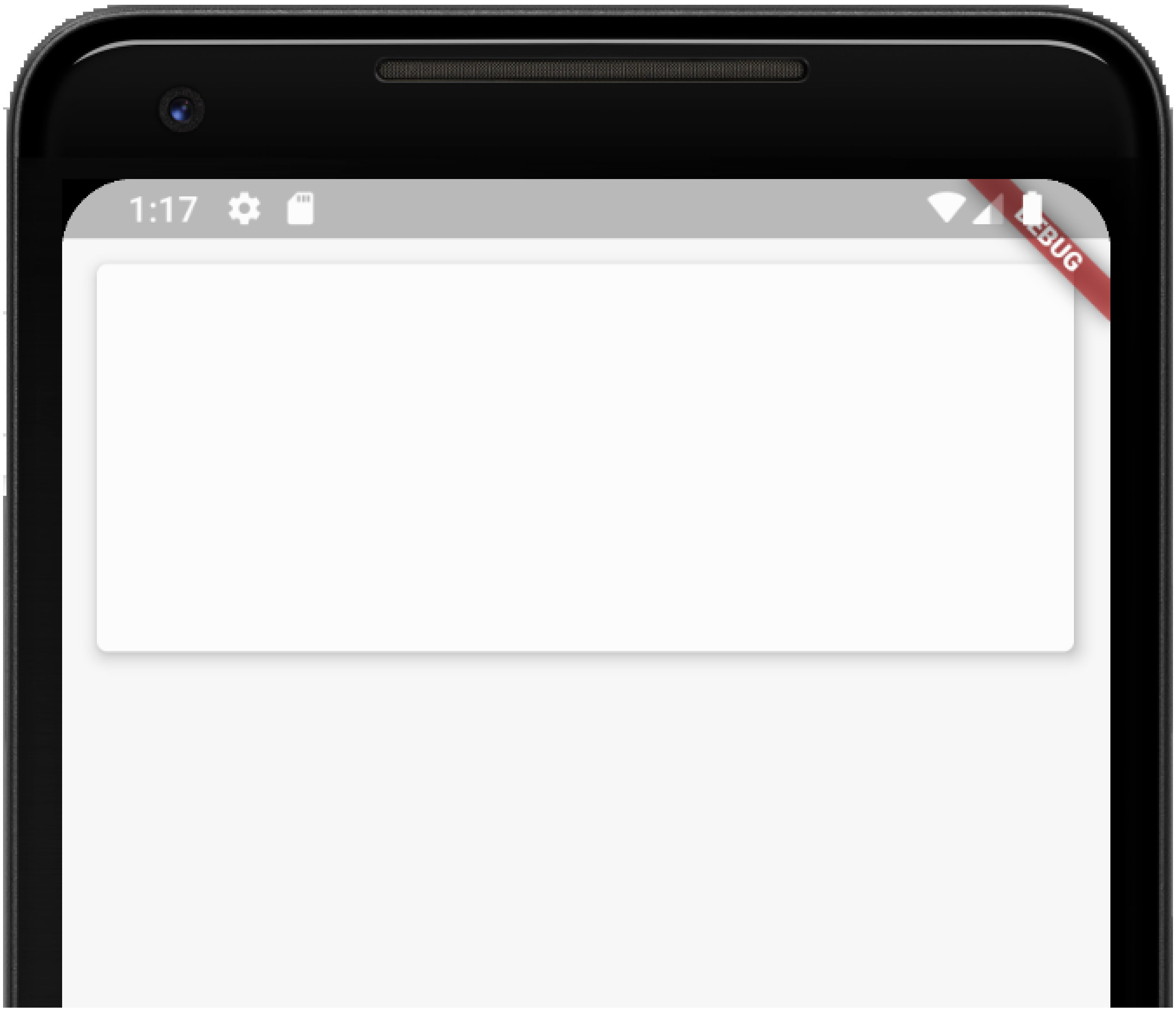
Step 2: Basic card creation

Now that we are done with basic setup, let's dive right into the card layout structure.

Replace the Text widget code with this code:

```
Scaffold(
 body: Container(
   padding: EdgeInsets.fromLTRB(10,10,10,0),
   height: 220,
   width: double.maxFinite,
   child: Card(
     elevation: 5,
   ),
 ),
```

Hurray!! Your card is now ready. You should see the following screen:



Step 3: Adding content to your card

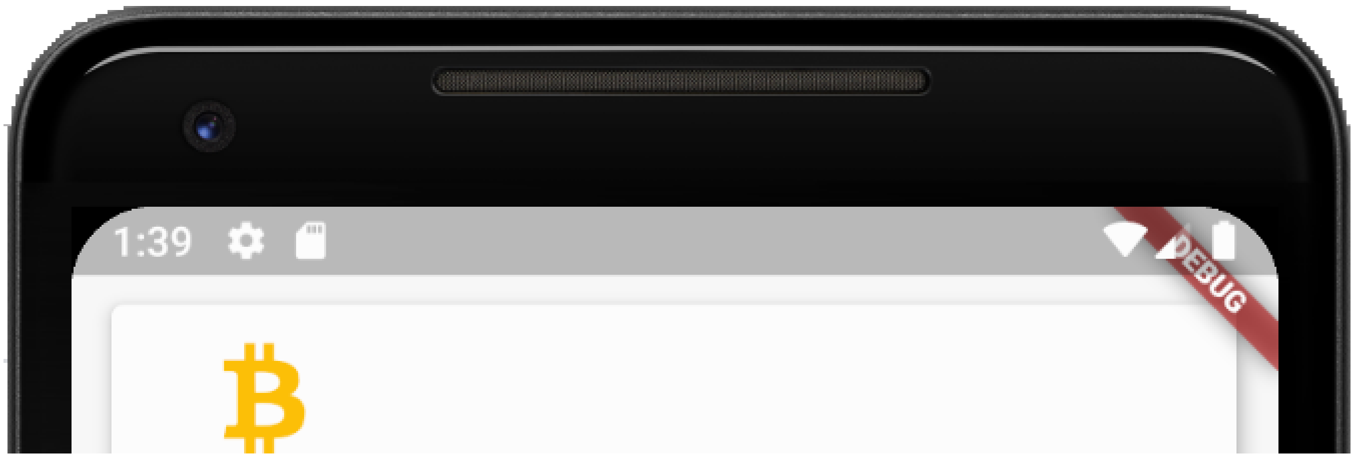
Everything seems simple so far. We have the basic card layout for us. Now we need to add content to it and align to accordingly.

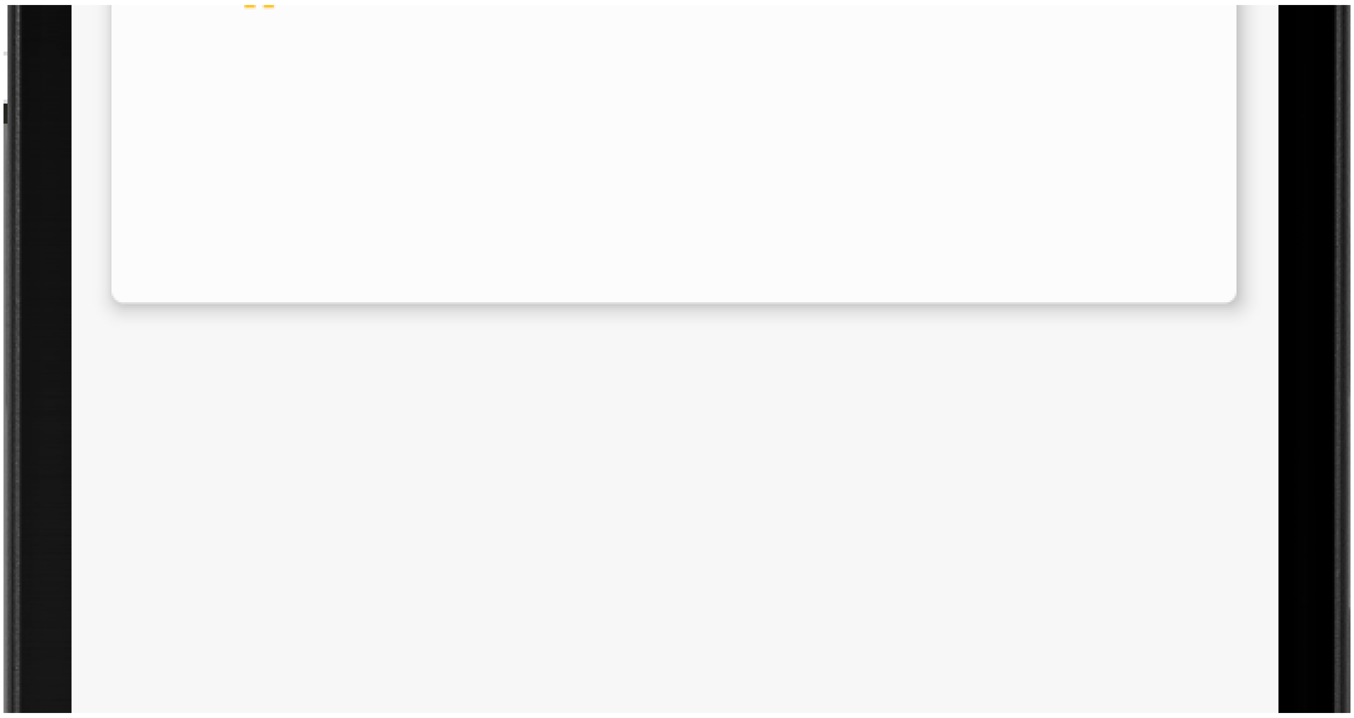
Let's start playing around. We are going to use **Stack** and **Align** widgets for positioning the content inside the card. You can read more about Stack here: https://api.flutter.dev/flutter/widgets/Stack-class.html.

Add this code under the **Card widget**:

```
child: Padding(
 padding: EdgeInsets.all(7),
 child: Stack(
 children: <Widget>[
   Align(
     alignment: Alignment.centerRight,
     child: Stack(
       children: <Widget>[
         Padding(
             padding: const EdgeInsets.only(left: 10, top: 5),
             child: Column(
               children: <Widget>[
                 Row(
                   children: <Widget>[
                     Padding(
                       padding: const EdgeInsets.only(left: 15.0),
                       child: Align(
                           alignment: Alignment.centerLeft,
                           child: Icon(
                             CryptoFontIcons.BTC,
                             color: Colors.amber,
                             size: 40,
                           )),
                     )
                   ],
                 )
               ],
             ))
       ],
     ),
   )
 ]),
),
```

Your output should look like this:





We just added an icon under the Stack. Now we will keep adding the rest of the content with suitable alignments.

To simplify the code, we will be moving each component code into a widget method and call that method inside the **Stack** Widget.

Like this:

```
Widget cryptoIcon(){
return Padding(
…
);
}
```

After adding all the components, the dashboard.dart file looks like this:

```
import 'package:crypto_font_icons/crypto_font_icons.dart';
import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';
import 'package:typicons_flutter/typicons.dart';

class Dashboard extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
 return MaterialApp(
     home: Scaffold(
         body: Container(
   padding: EdgeInsets.fromLTRB(10, 30, 10, 10),
   height: 220,
   width: double.maxFinite,
   child: Card(
     elevation: 5,
     child: Padding(
       padding: EdgeInsets.all(7),
       child: Stack(children: <Widget>[
         Align(
           alignment: Alignment.centerRight,
           child: Stack(
             children: <Widget>[
               Padding(
                   padding: const EdgeInsets.only(left: 10, top:
5),
                   child: Column(
                     children: <Widget>[
                       Row(
                         children: <Widget>[
                           cryptoIcon(),
                           SizedBox(
                             height: 10,
                           ),
                           cryptoNameSymbol(),
                           Spacer(),
                           cryptoChange(),
                           SizedBox(
                             width: 10,
                           ),
                           changeIcon(),
                           SizedBox(
                             width: 20,
                           )
                         ],
                       ),
                       Row(
                         children: <Widget>[cryptoAmount()],
                       )
                     ],
                   ))
             ],
           ),
         )
       ]),
     ),
   ),
 )));
 }

 Widget cryptoIcon() {
   return Padding(
     padding: const EdgeInsets.only(left: 15.0),
     child: Align(
         alignment: Alignment.centerLeft,
         child: Icon(
          CryptoFontIcons.BTC,
          color: Colors.amber,
          size: 40,
        )),
    );
 }

 Widget cryptoNameSymbol() {
   return Align(
     alignment: Alignment.centerLeft,
     child: RichText(
       text: TextSpan(
         text: 'Bitcoin',
         style: TextStyle(
             fontWeight: FontWeight.bold, color: Colors.black,
fontSize: 20),
         children: <TextSpan>[
           TextSpan(
               text: '\nBTC',
               style: TextStyle(
                   color: Colors.grey,
                   fontSize: 15,
                   fontWeight: FontWeight.bold)),
         ],
       ),
     ),
   );
  }

Widget cryptoChange() {
  return Align(
    alignment: Alignment.topRight,
    child: RichText(
      text: TextSpan(
        text: '+3.67%',
        style: TextStyle(
            fontWeight: FontWeight.bold, color: Colors.green,
fontSize: 20),
        children: <TextSpan>[
        TextSpan(
            text: '\n+202.835',
            style: TextStyle(
                color: Colors.green,
                fontSize: 15,
                fontWeight: FontWeight.bold)),
      ],
    ),
  ),
);
}

Widget changeIcon() {
  return Align(
      alignment: Alignment.topRight,
      child: Icon(
        Typicons.arrow_sorted_up,
        color: Colors.green,
        size: 30,
      ));
}

Widget cryptoAmount() {
  return Align(
    alignment: Alignment.centerLeft,
    child: Padding(
      padding: const EdgeInsets.only(left: 20.0),
      child: Row(
        children: <Widget>[
          RichText(
            textAlign: TextAlign.left,
            text: TextSpan(
              text: '\n\$12.279',
              style: TextStyle(
                color: Colors.grey,
                fontSize: 35,
              ),
              children: <TextSpan>[
                TextSpan(
                    text: '\n0.1349',
                    style: TextStyle(
                        color: Colors.grey,
                        fontStyle: FontStyle.italic,
                        fontSize: 20,
                        fontWeight: FontWeight.bold)),
              ],
            ),
          ),
        ],
      ),
    ),
  );
 }
}
```

What did we do in the above code?

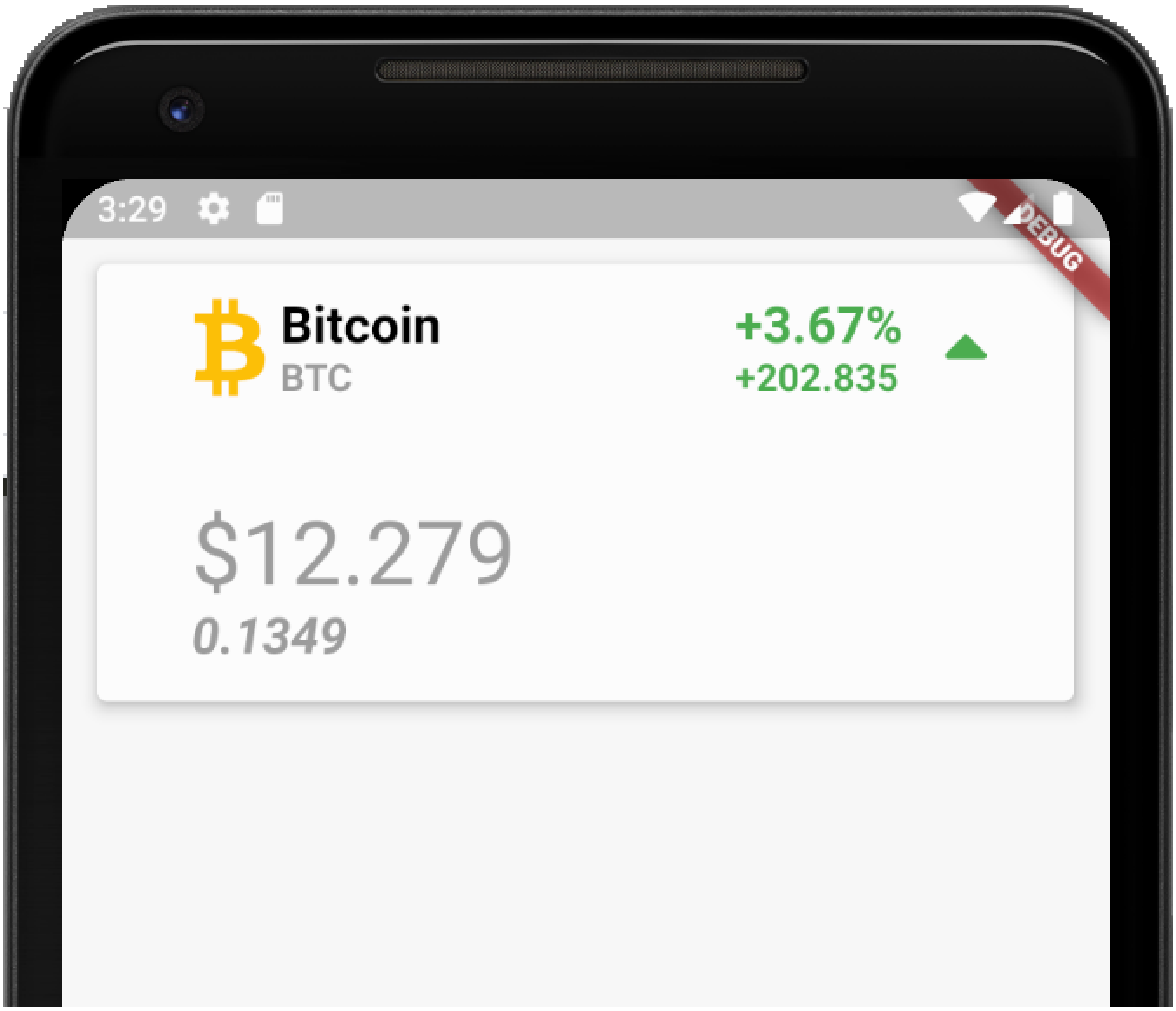
All we did was use **Align**, **RichText**, **Row** and **Icon** widgets.

The **Align** widget is used to Align the contents of the card properly.

The **RichText** widget is similar to the Text widget, but using RichText widget we can apply different styles to a Text. In the Text widget, the same style will be applied to the entire Text.

As the name suggests, a **Row** widget is used to render all its children in a single row. An **Icon** widget is used to render icons.

Once you run the code, your code should look like this:



Step 4: Add data into a JSON file and import it in the code.

We have the basic card layout now. But our aim is to build a list of cards and parameterize the code we built. For that, we need to have some data.

Create a new dart file *crypto_data.dart* and include the code below in it. You can add more data as json objects to the array.

```
import 'package:crypto_font_icons/crypto_font_icons.dart';
import 'package:flutter/material.dart';
```

```
class CryptoData {
 static final getData = [
   {
     ‘name’: ‘Bitcoin’,
     ‘symbol’: ‘BTC’,
     ‘icon’: CryptoFontIcons.BTC,
     ‘iconColor’: Colors.orange,
     ‘change’: ‘+3.67%’,
     ‘changeValue’: ‘+202.835’,
     ‘changeColor’: Colors.green,
     ‘value’: ‘\$12.279’,
   },
   {
     ‘name’: ‘Ethereum’,
     ‘symbol’: ‘ETH’,
     ‘icon’: CryptoFontIcons.ETH,
     ‘iconColor’: Colors.black,
     ‘change’: ‘+5.2%’,
     ‘changeValue’: ‘25.567’,
     ‘changeColor’: Colors.green,
     ‘value’: ‘\$7.809’
   },
 ];
}
```

Now we need to import this data into a variable in the dashboard file.

Include the below code to do the same.

```
var cryptoData = CryptoData.getData;
```

Step 5: The final step! Time to include the Card widget in a ListView and parameterize the widget methods

This step is pretty simple. We simply wrap the card component code inside a **ListView** builder and set a length to it.

We get this length from the **cryptoData** variable. We then parameterize all widget methods to bind each object data to the widgets.

The item Builder code looks like this:

```
Container(
             child: Column(
             mainAxisSize: MainAxisSize.max,
             mainAxisAlignment: MainAxisAlignment.start,
             children: <Widget>[
               Expanded(
                 child: ListView.builder(
                // scrollDirection: Axis.horizontal,
                     itemCount: cryptoData.length,
                     itemBuilder: (context, index) {
…
…
}),
 ),
 ],
 ),
)
```

We included the ListView builder under a column so we can set the size of the painting boundary to the maximum. The card code will be included inside the **itemBuilder** Method.

To parameterize the data widgets, we can access the object properties using `cryptoData[index][‘property’]` .

The *index* parameter takes care of holding the current object index while iterating through the **cryptoData** array.

After including all the object properties, you can add a top border for each card. In the current code, we have Padding widget as the child widget of **Card**.

Now wrap the **Padding** widget with a container and include the code below to render the border for each card dynamically.

```
decoration: BoxDecoration(
 border: Border(
   top: BorderSide(
       width: 2.0, color: cryptoData[index][‘iconColor’]),
 ),
 color: Colors.white,
),
```

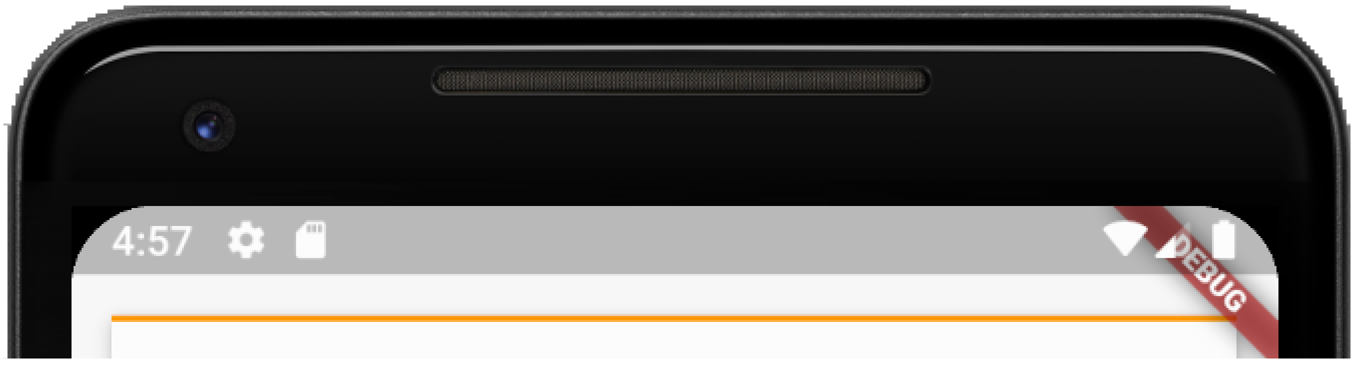
This is the final code for the Dashboard widget:

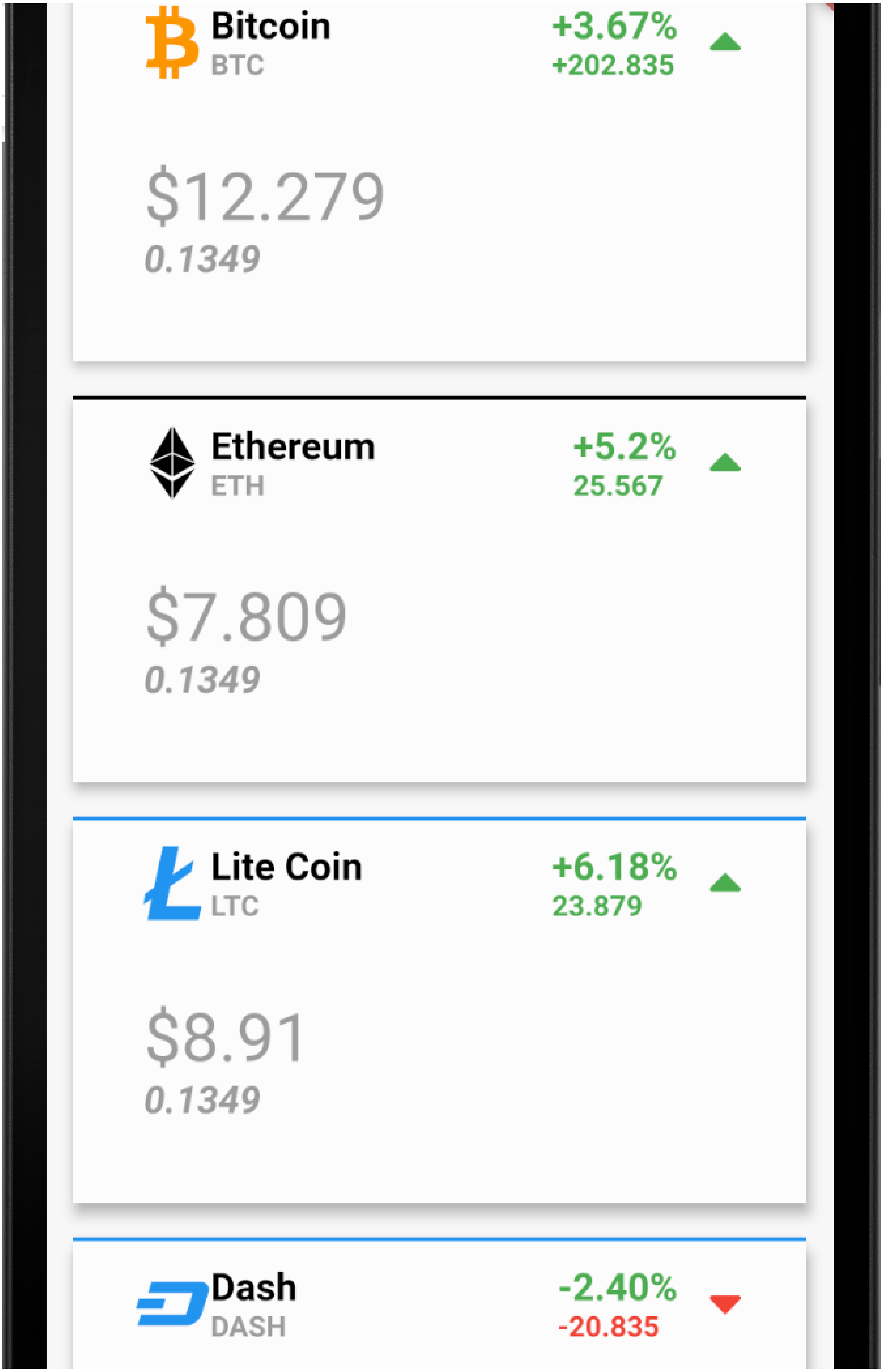
```
import ‘package:crypto_font_icons/crypto_font_icons.dart’;
import ‘package:flutter/cupertino.dart’;
import ‘package:flutter/material.dart’;
import ‘package:typicons_flutter/typicons.dart’;

import ‘crypto_data.dart’;

class Dashboard extends StatelessWidget {
 var cryptoData = CryptoData.getData;

 @override
 Widget build(BuildContext context) {
   return MaterialApp(
       home: Scaffold(
           body: Container(
             child: Column(
               mainAxisSize: MainAxisSize.max,
               mainAxisAlignment: MainAxisAlignment.start,
               children: <Widget>[
                 Expanded(
                   child: ListView.builder(
// scrollDirection: Axis.horizontal,
                   itemCount: cryptoData.length,
                   itemBuilder: (context, index) {
                     return Container(
                       padding: EdgeInsets.fromLTRB(10, 10, 10,
0),
                       height: 220,
                       width: double.maxFinite,
                       child: Card(
                         elevation: 5,
                         child: Container(
                           decoration: BoxDecoration(
                             border: Border(
                               top: BorderSide(
                                   width: 2.0, color:
cryptoData[index][‘iconColor’]),
                             ),
                             color: Colors.white,
                           ),
                           child: Padding(
                             padding: EdgeInsets.all(7),
                             child: Stack(children: <Widget>[
                               Align(
                                 alignment: Alignment.centerRight,
                                 child: Stack(
                                   children: <Widget>[
                                     Padding(
                                         padding: const
EdgeInsets.only(left: 10, top: 5),
                                         child: Column(
                                           children: <Widget>[
                                             Row(
                                               children: <Widget>[
                                                 cryptoIcon(cryp
toData[index]),
                                               SizedBox(
                                                 height: 10,
                                               ),
                                               cryptoNameSymbol(cr
yptoData[index]),
                                               Spacer(),
                                               cryptoChange(cr
yptoData[index]),
                                               SizedBox(
                                                 width: 10,
                                               ),
                                               changeIcon(cr
yptoData[index]),
                                               SizedBox(
                                                 width: 20,
                                               )
                                             ],
                                           ),
                                           Row(
                                           children: <Widget>[
                                             cryptoAmount(cr
yptoData[index])
                                           ],
                                         )
                                       ],
                                      ))
                                 ],
                               ),
                             )
                           ]),
                         ),
                       ),
                     ),
                   );
                 }),
           ),
         ],
       ),
     )));
 }

Widget cryptoIcon(data) {
  return Padding(
    padding: const EdgeInsets.only(left: 15.0),
    child: Align(
        alignment: Alignment.centerLeft,
        child: Icon(
          data['icon'],
          color: data['iconColor'],
          size: 40,
        )),
  );
 }
```

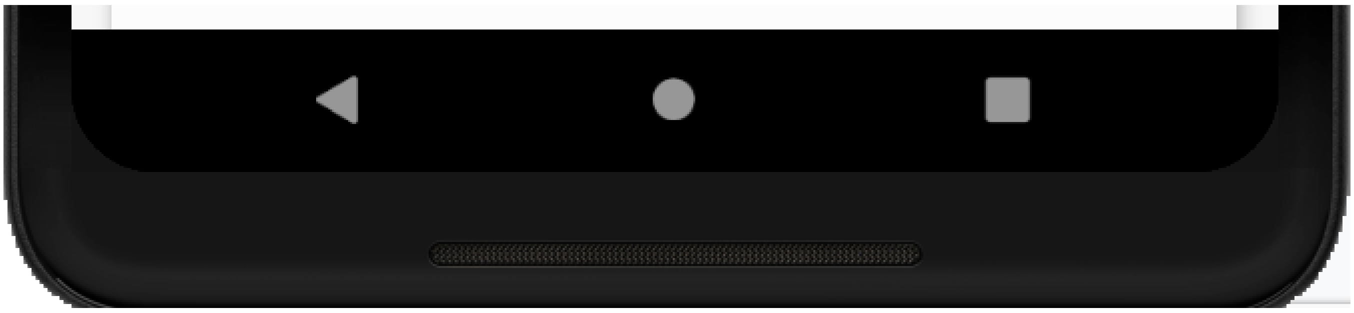
```
Widget cryptoNameSymbol(data) {
  return Align(
    alignment: Alignment.centerLeft,
    child: RichText(
      text: TextSpan(
        text: '${data['name']}',
        style: TextStyle(
            fontWeight: FontWeight.bold, color: Colors.black,
fontSize: 20),
        children: <TextSpan>[
          TextSpan(
              text: '\n${data['symbol']}',
              style: TextStyle(
                  color: Colors.grey,
                  fontSize: 15,
                  fontWeight: FontWeight.bold)),
          ],
        ),
      ),
    );
 }

Widget cryptoChange(data) {
  return Align(
    alignment: Alignment.topRight,
    child: RichText(
      text: TextSpan(
        text: '${data['change']}',
        style: TextStyle(
            fontWeight: FontWeight.bold, color: Colors.green,
fontSize: 20),
        children: <TextSpan>[
          TextSpan(
              text: '\n${data['changeValue']}',
              style: TextStyle(
                  color: data['changeColor'],
                  fontSize: 15,
                  fontWeight: FontWeight.bold)),
         ],
       ),
     ),
   );
 }

Widget changeIcon(data) {
  return Align(
      alignment: Alignment.topRight,
      child: data['change'].contains('-')
          ? Icon(
        Typicons.arrow_sorted_down,
        color: data['changeColor'],
        size: 30,
      )
          : Icon(
        Typicons.arrow_sorted_up,
        color: data['changeColor'],
        size: 30,
      ));
 }

Widget cryptoAmount(data) {
  return Align(
  alignment: Alignment.centerLeft,
  child: Padding(
    padding: const EdgeInsets.only(left: 20.0),
    child: Row(
      children: <Widget>[
        RichText(
          textAlign: TextAlign.left,
          text: TextSpan(
            text: '\n${data['value']}',
            style: TextStyle(
              color: Colors.grey,
              fontSize: 35,
            ),
            children: <TextSpan>[
              TextSpan(
                  text: '\n0.1349',
                  style: TextStyle(
                      color: Colors.grey,
                      fontStyle: FontStyle.italic,
                      fontSize: 20,
                      fontWeight: FontWeight.bold)),
            ],
          ),
        ),
      ],
    ),
  ),
);
}
}
```

Done !!! You are now ready to view the final output.

The following screen will be the output once you run the new code.




Bitcoin
BTC
+3.67%
+202.835
$12.279
0.1349
Ethereum
ETH
+5.2%
25.567
$7.809
0.1349
Lite Coin
LTC
+6.18%
23.879
$8.91
0.1349
Dash
DASH
-2.40%
-20.835



Congratulations!! You have now built a basic card list in Flutter with some decent styling. Now that you have finished this first bit why not build a Flutter app bar dashboard in 10 minutes or less? Check it out here:

How to Build a Flutter™ App Bar Dashboard in 10 Minutes or Less

In this second part of our FlutterIn5Series, we are going to build an app bar dashboard in 5 simple steps

DLT Labs on medium.com

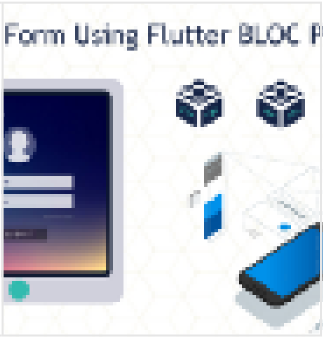
Or,

If you want to try something more challenging, how about building a simple login form in Flutter.

How to create a simple login form in Flutter™ using BLoC pattern

Flutter has many benefits. We show you how to make a simple login form using Flutter's BLoC pattern.

DLT Labs on medium.com



• • •

DLT Labs is a trademark of DLT Global, Inc. Flutter™ is a trademark of Google LLC and its use here does not indicate endorsement or affiliation.

Author —Arun Kashyap Karri, DLT Labs™

About the Author: Arun Kashyap Karri is an active learner and an enthusiastic Web / Mobile Developer. He has worked on Angular, Node.js, Flutter, RPA, Chatbots and few more technologies. At DLT Labs, Arun has worked on applications such as Element DB, DL Certify(Mobile) and We Agree.

. . .

DLT LABS™
NEXT GENERATION MODULAR SOLUTIONS