Name: Yukta Bhatia
RollNo: 07
Class: D15A

Experiment-2

Aim: To design Flutter UI by including common widgets.

Theory:

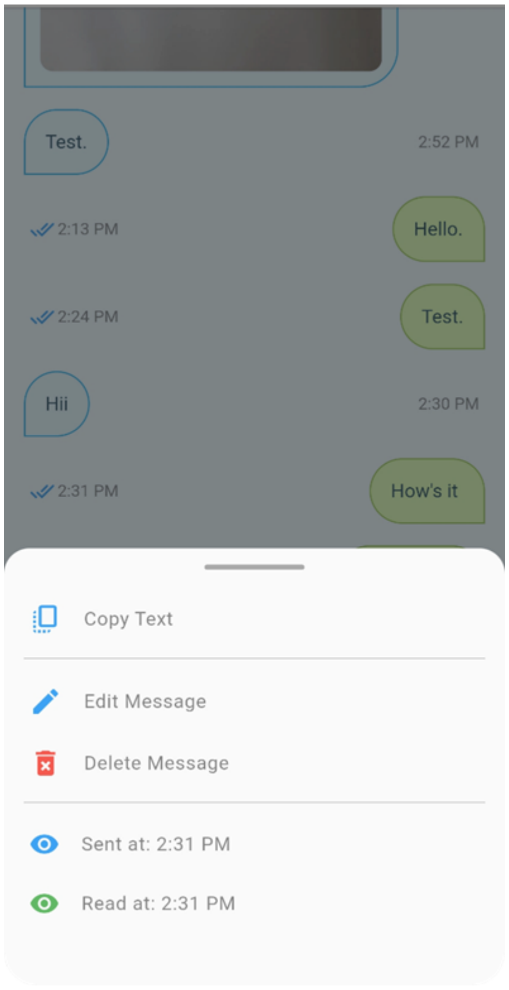
Flutter widgets are the building blocks of a Flutter app’s user interface. They are the basic visual elements developers use to create user interfaces and define the app’s functionality.

A Flutter widget can be defined as a self-contained, reusable piece of code that describes how part of the user interface should be displayed. Widgets can be thought of as Lego blocks, which can be combined and arranged in many different ways to create complex user interfaces.

Code:

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

class _MessageCardState extends State<MessageCard> {
  @override
  Widget build(BuildContext context) {
    bool isMe = APIs.user.uid == widget.message.fromId;
    return InkWell(
        onLongPress: () {
          _showBottomSheet(isMe);
        },
   }

  void _showBottomSheet(bool isMe) {
    showModalBottomSheet(
        context: context,
        shape: const RoundedRectangleBorder(
            borderRadius: BorderRadius.only(
                topLeft: Radius.circular(20), topRight: Radius.circular(20))),
        builder: (_) {
          return ListView(
shrinkWrap: true,
children: [
  //black divider
  Container(
    height: 4,
    margin: EdgeInsets.symmetric(
        vertical: mq.height * .015, horizontal: mq.width * .4),
    decoration: BoxDecoration(
        color: Colors.grey, borderRadius: BorderRadius.circular(8)),
  ),

  widget.message.type == Type.text
      ?
      //copy option
      _OptionItem(
          icon: const Icon(Icons.copy_all_rounded,
              color: Colors.blue, size: 26),
          name: 'Copy Text',
          onTap: () async {
            await Clipboard.setData(
                    ClipboardData(text: widget.message.msg))
                .then((value) {
              //for hiding bottom sheet
              Navigator.pop(context);

              Dialogs.showSnackbar(context, 'Text Copied!');
            });
          })
      :
      //save option
      _OptionItem(
          icon: const Icon(Icons.download_rounded,
              color: Colors.blue, size: 26),
          name: 'Save Image',
          onTap: () async {
            try {
              log('Image Url: ${widget.message.msg}');
              await GallerySaver.saveImage(widget.message.msg,
                      albumName: 'We Chat')
                  .then((success) {
                //for hiding bottom sheet
                Navigator.pop(context);
                if (success != null && success) {
                  Dialogs.showSnackbar(
                context, 'Image Successfully Saved!');
          }
        });
      } catch (e) {
        log('ErrorWhileSavingImg: $e');
      }
    }),

//separator or divider
if (isMe)
  Divider(
    color: Colors.black54,
    endIndent: mq.width * .04,
    indent: mq.width * .04,
  ),

//edit option
if (widget.message.type == Type.text && isMe)
  _OptionItem(
      icon: const Icon(Icons.edit, color: Colors.blue, size: 26),
      name: 'Edit Message',
      onTap: () {
        //for hiding bottom sheet
        Navigator.pop(context);

        _showMessageUpdateDialog();
      }),

//delete option
if (isMe)
  _OptionItem(
      icon: const Icon(Icons.delete_forever,
          color: Colors.red, size: 26),
      name: 'Delete Message',
      onTap: () async {
        await APIs.deleteMessage(widget.message).then((value) {
          //for hiding bottom sheet
          Navigator.pop(context);
        });
      }),

//separator or divider
Divider(
  color: Colors.black54,
              endIndent: mq.width * .04,
              indent: mq.width * .04,
            ),

            //sent time
            _OptionItem(
                icon: const Icon(Icons.remove_red_eye, color: Colors.blue),
                name:
                    'Sent At: ${MyDateUtil.getMessageTime(context: context, time:
widget.message.sent)}',
                onTap: () {}),

            //read time
            _OptionItem(
                icon: const Icon(Icons.remove_red_eye, color: Colors.green),
                name: widget.message.read.isEmpty
                    ? 'Read At: Not seen yet'
                    : 'Read At: ${MyDateUtil.getMessageTime(context: context, time:
widget.message.read)}',
                onTap: () {}),
          ],
        );
      });
  }

class _OptionItem extends StatelessWidget {
  final Icon icon;
  final String name;
  final VoidCallback onTap;

  const _OptionItem(
      {required this.icon, required this.name, required this.onTap});

  @override
  Widget build(BuildContext context) {
    return InkWell(
        onTap: () => onTap(),
        child: Padding(
          padding: EdgeInsets.only(
              left: mq.width * .05,
              top: mq.height * .015,
              bottom: mq.height * .015),
          child: Row(children: [
            icon,
            Flexible(
                child: Text('    $name',
                    style: const TextStyle(
                        fontSize: 15,
                        color: Colors.black54,
                        letterSpacing: 0.5)))
          ]),
        ));
  }
}
```

Output:



Test. 2:52 PM

2:13 PM Hello.

2:24 PM Test.

Hii 2:30 PM

2:31 PM How's it

Sent at: 2:31 PM

Read at: 2:31 PM