Aim - To include icons, images, and fonts in the Flutter app.

Theory -

1. Icons
   An icon is a graphic image representing an application or any specific entity containing meaning for the user. It can be selectable and non-selectable. For example, the company's logo is non-selectable. Sometimes it also contains a hyperlink to go to another page. It also acts as a sign in place of a detailed explanation of the actual entity.

   Flutter provides an Icon Widget to create icons in our applications. We can create icons in Flutter, either using inbuilt icons or custom icons. Flutter provides the list of all icons in the Icons class. In this article, we are going to learn how to use Flutter icons in the application.

2. Image
   In this section, we are going to see how we can display images in Flutter. When you create an app in Flutter, it includes both code and assets (resources). An asset is a file, which is bundled and deployed with the app and is accessible at runtime. The asset can include static data, configuration files, icons, and images. Flutter supports many image formats, such as JPEG, WebP, PNG, GIF, animated WebP/GIF, BMP, and WBMP. Displaying images is the fundamental concept of most mobile apps. Flutter has an Image widget that allows displaying of different types of images in the mobile application.

3. Text
   A Text is a widget in Flutter that allows us to display a string of text with a single line in our application. Depending on the layout constraints, we can break the string across multiple lines, or might all be displayed on the same line. If we do not specify any styling to the text widget, it will use the closest DefaultTextStyle class style. This class does not have any explicit style. In this article, we are going to learn how to use a Text widget and how to style it in our application.

4. Buttons
   Buttons are the graphical control element that provides a user to trigger an event such as taking actions, making choices, searching things, and many more. They can be placed anywhere in our UI like dialogs, forms, cards, toolbars, etc.
   Buttons are the Flutter widgets, which are a part of the material design library. Flutter provides several types of buttons that have different shapes, styles, and features.

   Features of Buttons

   - The standard features of a button in Flutter are given below:
   - We can easily apply themes to buttons, shapes, colors, animation, and behavior.
   - We can also theme icons and text inside the button.
   - Buttons can be composed of different child widgets with different characteristics.

Types of Flutter Buttons

Following are the different types of buttons available in Flutter:

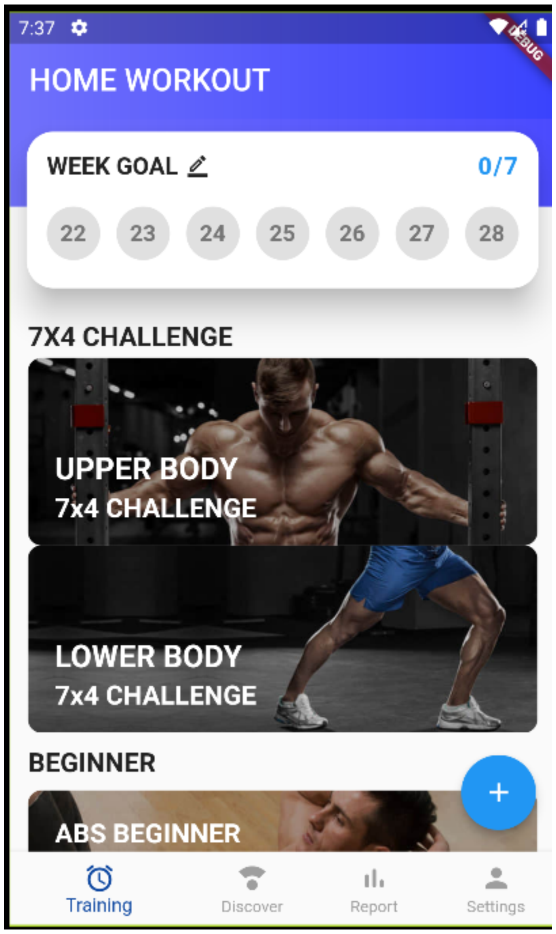
1. Flat Button
2. Raised Button
3. Floating Button
4. Drop Down Button
5. Icon Button
6. Inkwell Button
7. PopupMenu Button
8. Outline Button

Output -

```
 floatingActionButton: FloatingActionButton(
        onPressed: () {},
        child: Icon(Icons.add),
      ),
      bottomNavigationBar: BottomNavigationBar(
        currentIndex: _currentIndex,
        showSelectedLabels: true,
        showUnselectedLabels: true,
        selectedItemColor: Colors.blue[900],
        unselectedItemColor: Colors.grey,
        selectedIconTheme: IconThemeData(color: Colors.blue[900]),
        onTap: ((value) => setState(() {
              _currentIndex = value;
            })),
        items: [
          BottomNavigationBarItem(
            icon: Icon(
              Icons.alarm,
            ),
            label: 'Training',
          ),
          BottomNavigationBarItem(
            icon: Icon(
              Icons.compass_calibration,
            ),
            label: 'Discover',
          ),
          BottomNavigationBarItem(
            icon: Icon(
              Icons.bar_chart_rounded,
            ),
          label: 'Report',
        ),
        BottomNavigationBarItem(
          icon: Icon(
            Icons.person,
          ),
          label: 'Settings',
        ),
      ],
    ),
  );
 }
}

class homePageChallengeCard extends StatelessWidget {
 late List details;
 homePageChallengeCard(this.details, {Key? key}) : super(key: key);

 @override
 Widget build(BuildContext context) {
  return Column(
    children: [
     SizedBox(
       height: 140,
       child: Stack(children: [
         Positioned.fill(
           child: ClipRRect(
             borderRadius: BorderRadius.circular(10),
             child: Image(
               image: AssetImage('assets/${details[0]}'),
               fit: BoxFit.fitWidth,
             ),
           ),
         ),
         Positioned(
             top: 70,
             left: 20,
             child: Text(
               '${details[1]}',
               style: TextStyle(
                 color: Colors.white,
                 fontSize: 23,
                 fontWeight: FontWeight.bold,
               ),
              )),
            Positioned(
              top: 100,
              left: 20,
              child: Text(
                '7x4 CHALLENGE',
                style: TextStyle(
                  color: Colors.white,
                  fontSize: 20,
                  fontWeight: FontWeight.bold,
                ),
              )),
          ]),
        ),
      ],
    );
  }
}
```



Conclusion -
We have learned about icons, images, fonts, buttons, and other widgets in this experiment. We also implemented these widgets in our app and changed the design, style, and color as per our requirements.