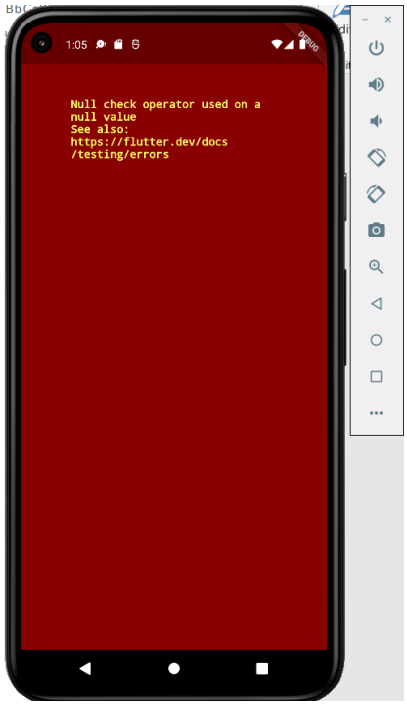
Lab-6

Link for assignment : https://github.com/Diya-Lad/SDP

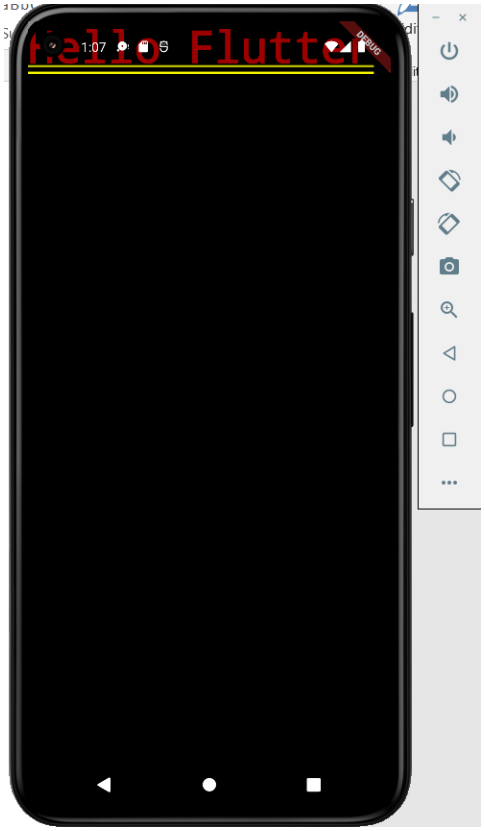
Code test 1:

```
void main() {
  runApp(MaterialApp());
}
```



Code test 2:

```
void main() {
  runApp(MaterialApp(
    home: Text('Hello Flutter'),
  ));
}
```



Code test 3:

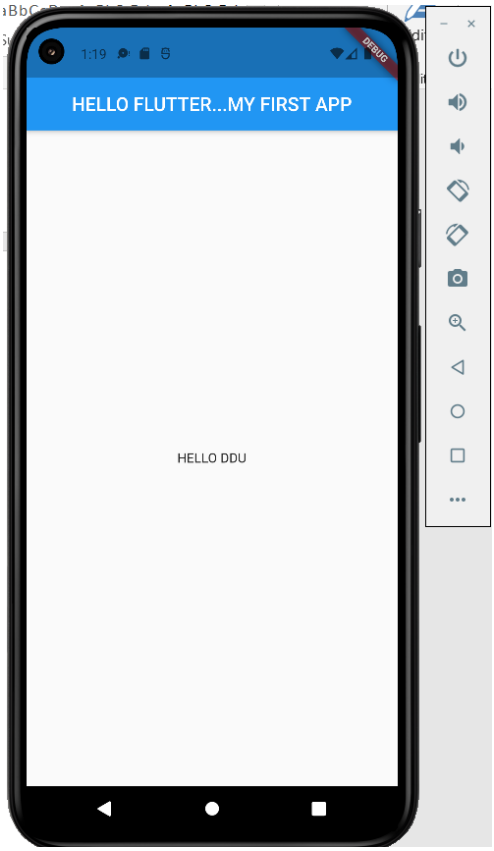
```
void main() {
  runApp(MaterialApp(
    home: Scaffold(
      appBar: AppBar(
        title: Text('HELLO FLUTTER...MY FIRST APP'),
        centerTitle: true,
      ),
    ),
  ));
}
```

- The Scaffold widget, from the Material library, provides a default app bar, a title, and a body property that holds the widget tree for the home screen. The widget subtree can be quite complex.

```
void main() {
  runApp(MaterialApp(
    home: Scaffold(
        appBar: AppBar(
          title: Text('HELLO FLUTTER...MY FIRST APP'),
          centerTitle: true,
```

```
        ),
        body: Center(
          child: Text('HELLO DDU'),
        )
    ),
  ));
}
```

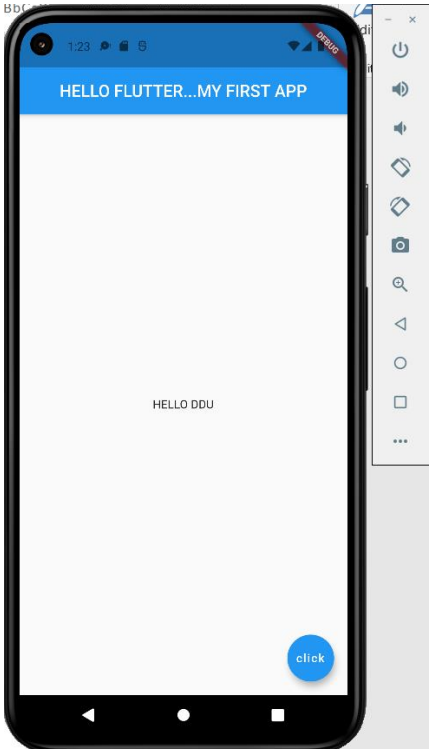
- The body for this example consists of a Center widget containing a Text child widget. The Center widget aligns its widget subtree to the center of the screen.




```
void main() {
  runApp(MaterialApp(
    home: Scaffold(
        appBar: AppBar(
          title: Text('HELLO FLUTTER...MY FIRST APP'),
          centerTitle: true,
        ),
        body: Center(
          child: Text('HELLO DDU'),
        ),
        floatingActionButton: FloatingActionButton(
          onPressed: () { },
          child: Text('click'),
        ),
```

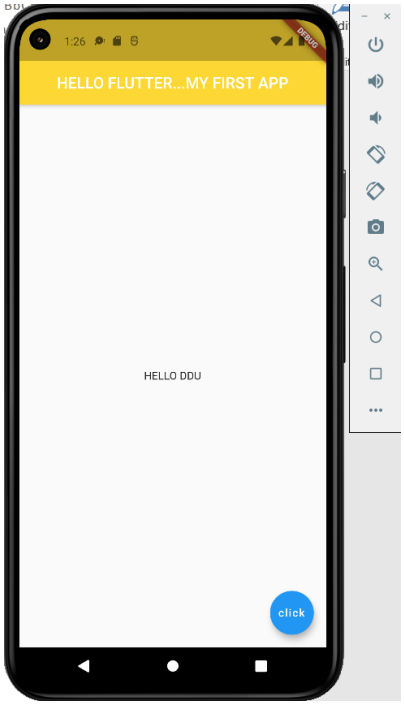
```
    ),
  ));
```

- A floating action button is a circular icon button that hovers over content to promote a primary action in the application.



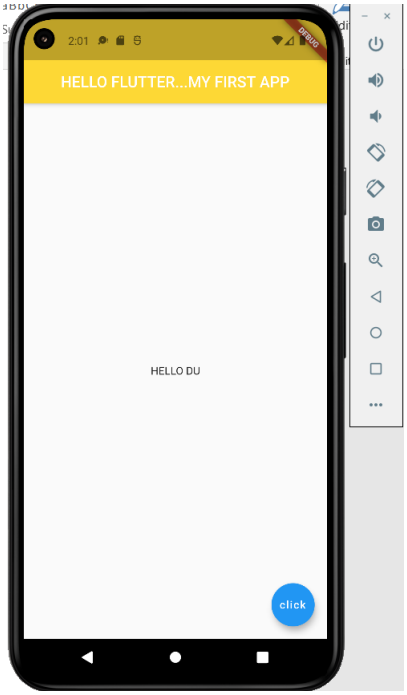
Code test 4: customization

```
void main() {
  runApp(MaterialApp(
    home: Scaffold(
        appBar: AppBar(
          title: Text('HELLO FLUTTER...MY FIRST APP'),
          backgroundColor: Colors.yellow[600],
          centerTitle: true,
        ),
        body: Center(
          child: Text('HELLO DDU'),
        ),
        floatingActionButton: FloatingActionButton(
          onPressed: () { },
          child: Text('click'),
        ),
    ),
  ));
}
```



Code test 5 : Inserting new font to project

```
void main() {
  runApp(MaterialApp(
    home: Scaffold(
        appBar: AppBar(
          title: Text('HELLO FLUTTER...MY FIRST APP'),
          backgroundColor: Colors.yellow[600],
          centerTitle: true,
        ),
        body: Center(
          child: Text('HELLO DU', style: TextStyle(
              fontFamily: 'Inconsolable'
          ),),
        ),
        floatingActionButton: FloatingActionButton(
          onPressed: () { },
          child: Text('click'),
        ),
    ),
  ));
}
```



Code test 6: Hot reloading through stateless widget

- Stateless widgets are immutable, meaning that their properties can't change—all values are final.
- Stateful widgets maintain state that might change during the lifetime of the widget.
    - Implementing a stateful widget requires at least two classes, a StatefulWidget that creates an instance of a State class.
    - The StatefulWidget object is, itself, immutable and can be thrown away and regenerated, but the State object persists over the lifetime of the widget.

➢ **Difference between Hot reload and Hot Restart**

○ Hot Reload :

- Hot Reload allows us to see the reflected change after bug fixes, building User interfaces and even adding certain features to the app without running your application afresh over and over again.

- Hot Reload performs very as compared to hot restart or default restart of flutter.
- We perform hot reload by using key ctlr+\.

- Hot Restart :

- Hot restart destroys the preserved State value and set them to their default.
- Hot Restart is slower than hot reload but faster than the default restart.
- We perform hot restart using ctrl+shift+\