Text

Description automatically generated

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
| **Assignment No** | 4 |

Assignment Number - 04

**Title :** Design Flutter Application for TODO list.

**Theory :**

**SuffixIcon:**

Suffix Icon : The suffix icon is constrained with a minimum size of 48px by 48px, but can be expanded beyond that. Anything larger than 24px will require additional padding to ensure it matches the Material Design spec of 12px padding between the right edge of the input and trailing edge of the prefix icon.



**Figure 1: Prefix & Suffix Icon**

**Asynchronous Code:**

Asynchronous operations let your program complete work while waiting for another operation to finish. Here are some common asynchronous operations:

* Fetching data over a network.
* Writing to a database.
* Reading data from a file.

Such asynchronous computations usually provide their result as a Future or, if the result has multiple parts, as a Stream. These computations introduce asynchrony into a program. To accommodate that initial asynchrony, other plain Dart functions also need to become asynchronous.

To interact with these asynchronous results, you can use the async and await keywords. Most asynchronous functions are just async Dart functions that depend, possibly deep down, on an inherently asynchronous computation.

* **synchronous operation:** A synchronous operation blocks other operations from executing until it completes.
* **synchronous function**: A synchronous function only performs synchronous operations.
* **asynchronous operation:** Once initiated, an asynchronous operation allows other operations to execute before it completes.
* **asynchronous function:** An asynchronous function performs at least one asynchronous operation and can also perform synchronous operations. A **Synchronous function is a function that does not return until the work is completed or has failed.**

**async and await :**

The async and await keywords provide a declarative way to define asynchronous functions and use their results. Remember these two basic guidelines when using async and await.

• To define an async function, add async before the function body:

• The await keyword works only in async functions.

**Source Code:-**

import 'package:flutter/material.dart';

void main() => runApp(const MyApp());

class MyApp extends StatelessWidget {

const MyApp({super.key});

@override

Widget build(BuildContext context) {

return const MaterialApp(

title: 'Task List',

home: WorkList(),

);

}

}

class Work {

final String name;

final String work;

final String date;

Work(this.name, this.work, this.date);

}

class WorkList extends StatefulWidget {

const WorkList({super.key});

@override

// ignore: library\_private\_types\_in\_public\_api

\_WorkListState createState() => \_WorkListState();

}

class \_WorkListState extends State<WorkList> {

final TextEditingController \_nameController = TextEditingController();

final TextEditingController \_workController = TextEditingController();

final TextEditingController \_dateController = TextEditingController();

final List<Work> \_works =

[]; void \_addWork() {

setState(() {

final name = \_nameController.text;

final work = \_workController.text;

final date = \_dateController.text;

final newWork = Work(name, work, date);

\_works.add(newWork);

\_nameController.clear();

\_workController.clear();

\_dateController.clear();

});

}

@override

Widget build(BuildContext context)

{ return Scaffold( appBar:

AppBar(

title: const Text('TO-DO LIST'),

),

body: Column(

children: <Widget>[

TextField(

controller: \_nameController,

decoration: const InputDecoration(

labelText: 'TODO Task',

icon:Icon(Icons.drive\_file\_rename\_outline\_outlined),

),

),

TextField(

controller: \_workController,

decoration: const InputDecoration(

labelText: 'Description of Task',

icon:Icon(Icons.work),

),

),

TextField(

controller: \_dateController,

decoration: InputDecoration(

labelText: ' DATE',

icon: const Icon(Icons.date\_range\_outlined),

suffixIcon: IconButton(

icon: const Icon(Icons.calendar\_today),

onPressed: ()

async {

final DateTime? date = await showDatePicker(

context: context,

initialDate: DateTime.now(),

firstDate: DateTime(2022),

lastDate: DateTime.now(),

);

if (date != null) {

\_dateController.text = date.toString().substring(0, 10);

}

},

),

), ),

const SizedBox(height: 50.0),

ElevatedButton( onPressed: \_addWork, child: const

Text('Submit'),

),

Expanded(

child: ListView.builder(

itemCount: \_works.length,

itemBuilder: (BuildContext context, int index)

{ final work = \_works[index];

return ListTile( title: Text(work.name),

subtitle: Column(

crossAxisAlignment: CrossAxisAlignment.start,

children: [

Text(work.work),

Text(work.date),

],

),

);

},

),

),

],

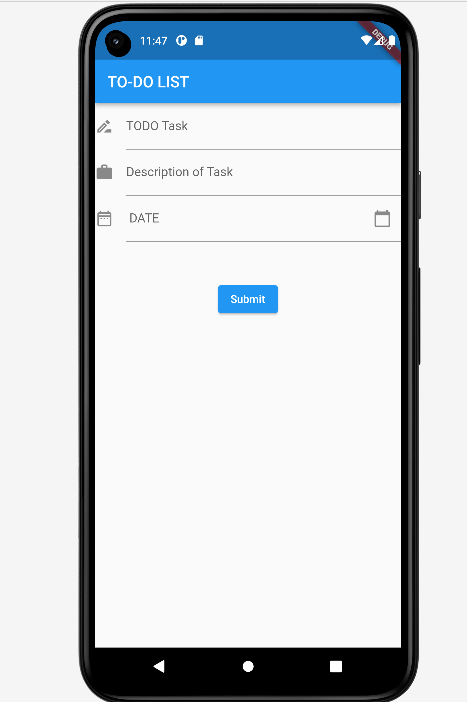
),

);

}

}

**Output:**



**Conclusion *:***