**Assessment:**

**Lab Task 1: Build a Shopping List App Using Flutter**

**Objective**

Create a simple Shopping List app where users can:

* Add items to the list
* Mark items as purchased
* Delete items from the list

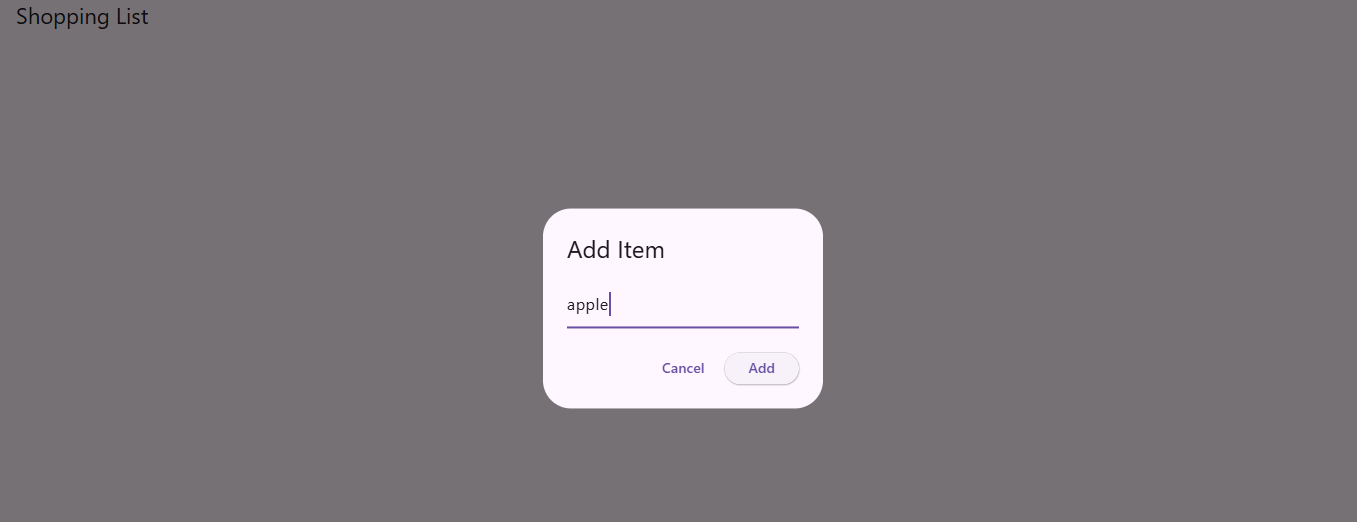
**Requirements**

1. Use StatefulWidget to manage the list.
2. Each item should have a name and a purchased status.
3. Use a TextField in a dialog to add new items.
4. Display items in a ListView.
5. Tap an item or use a checkbox to mark it as purchased (with strikethrough).
6. Include a delete icon to remove items.
7. Use a FloatingActionButton to add items.

Main.dart:

| import 'package:flutter/material.dart';  void main() {  runApp(ShoppingListApp());  }  class ShoppingListApp extends StatelessWidget {  @override  Widget build(BuildContext context) {  return MaterialApp(  title: 'Shopping List',  debugShowCheckedModeBanner: false,  home: ShoppingListScreen(),  );  }  }  // Item model  class ShoppingItem {  String name;  bool isPurchased;  ShoppingItem({required this.name, this.isPurchased = false});  }  // Main Screen  class ShoppingListScreen extends StatefulWidget {  @override  \_ShoppingListScreenState createState() => \_ShoppingListScreenState();  }  class \_ShoppingListScreenState extends State<ShoppingListScreen> {  final List<ShoppingItem> \_items = [];  void \_addItem(String name) {  setState(() {  \_items.add(ShoppingItem(name: name));  });  }  void \_showAddItemDialog() {  final TextEditingController \_controller = TextEditingController();  showDialog(  context: context,  builder: (\_) => AlertDialog(  title: Text('Add Item'),  content: TextField(  controller: \_controller,  autofocus: true,  decoration: InputDecoration(hintText: 'Enter item name'),  ),  actions: [  TextButton(  onPressed: () {  Navigator.pop(context);  },  child: Text('Cancel'),  ),  ElevatedButton(  onPressed: () {  if (\_controller.text.trim().isNotEmpty) {  \_addItem(\_controller.text.trim());  Navigator.pop(context);  }  },  child: Text('Add'),  ),  ],  ),  );  }  void \_togglePurchased(int index) {  setState(() {  \_items[index].isPurchased = !\_items[index].isPurchased;  });  }  void \_deleteItem(int index) {  setState(() {  \_items.removeAt(index);  });  }  Widget \_buildItemTile(int index) {  final item = \_items[index];  return ListTile(  leading: Checkbox(  value: item.isPurchased,  onChanged: (\_) => \_togglePurchased(index),  ),  title: Text(  item.name,  style: TextStyle(  decoration: item.isPurchased ? TextDecoration.lineThrough : null,  ),  ),  trailing: IconButton(  icon: Icon(Icons.delete, color: Colors.red),  onPressed: () => \_deleteItem(index),  ),  onTap: () => \_togglePurchased(index),  );  }  @override  Widget build(BuildContext context) {  return Scaffold(  appBar: AppBar(title: Text('Shopping List')),  body: \_items.isEmpty  ? Center(child: Text('No items. Tap + to add.'))  : ListView.builder(  itemCount: \_items.length,  itemBuilder: (context, index) => \_buildItemTile(index),  ),  floatingActionButton: FloatingActionButton(  onPressed: \_showAddItemDialog,  child: Icon(Icons.add),  ),  );  }  } |
| --- |

**OUTPUT:**

****

****

****

**Lab Task 2: Answer the following questions.**

Q1: What is the purpose of the Provider package in Flutter?

To manage and share state across widgets efficiently using the widget tree.

Q2: What is the difference between ChangeNotifierProvider and Provider?

Provider: Supplies static data.

ChangeNotifierProvider: Supplies data that notifies listeners when changed.

Q3: How do you update and listen to changes in state using Provider?

Use a class with ChangeNotifier, call notifyListeners() after updates, and listen using Consumer or context.watch().

Q4: Explain the role of Consumer in the Provider package.

Consumer rebuilds only the widget that depends on the provided data when it changes, optimizing performance.