**Assessment:**

**Task 1:** Create a Flutter app for managing a simple list of tasks. Use the Provider package to manage the list of tasks, where each task has a title, description, and a completion status.

Expected Output:

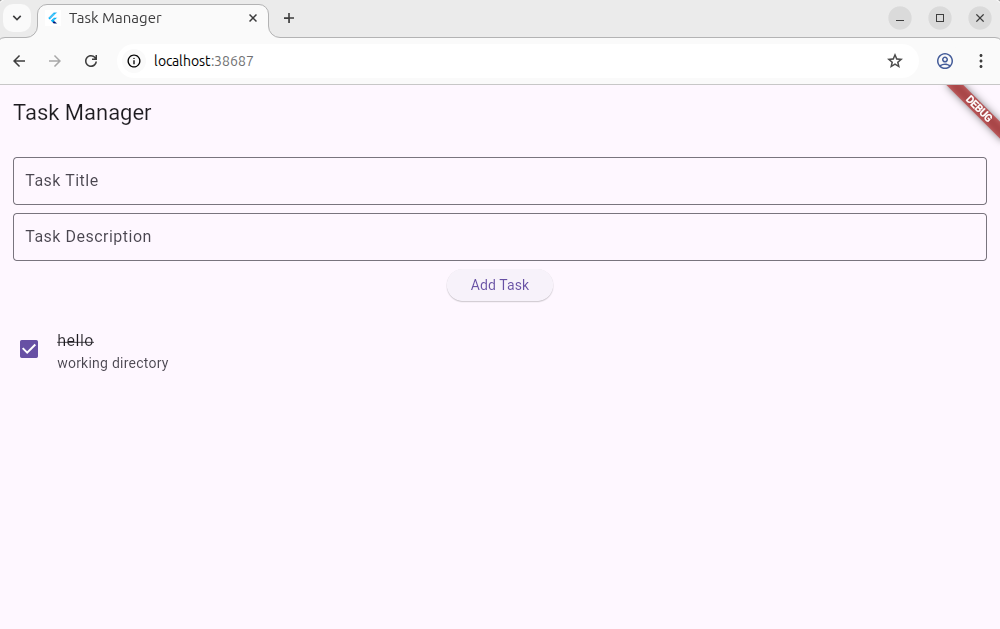
A basic task management app where:

* A list of tasks can be added and displayed.
* Each task has a checkbox to mark it as completed.
* The task list updates immediately upon adding or toggling tasks.

Main.dart:

|  |
| --- |
| // ignore\_for\_file: use\_super\_parameters  import 'package:flutter/material.dart';  import 'package:provider/provider.dart';  // Task model  class Task {  String title;  String description;  bool isCompleted;  Task({  required this.title,  required this.description,  this.isCompleted = false,  });  }  // Task provider for state management  class TaskProvider extends ChangeNotifier {  final List<Task> \_tasks = [];  List<Task> get tasks => \_tasks;  void addTask(String title, String description) {  \_tasks.add(Task(title: title, description: description));  notifyListeners();  }  void toggleTaskCompletion(int index) {  \_tasks[index].isCompleted = !\_tasks[index].isCompleted;  notifyListeners();  }  }  // Main app  void main() {  runApp(const TaskApp());  }  class TaskApp extends StatelessWidget {  const TaskApp({Key? key}) : super(key: key);    @override  Widget build(BuildContext context) {  return ChangeNotifierProvider(  create: (context) => TaskProvider(),  child: MaterialApp(  title: 'Task Manager',  theme: ThemeData(  primarySwatch: Colors.blue,  useMaterial3: true,  ),  home: const TaskListScreen(),  ),  );  }  }  // Task list screen  class TaskListScreen extends StatelessWidget {  const TaskListScreen({Key? key}) : super(key: key);  @override  Widget build(BuildContext context) {  final taskProvider = Provider.of<TaskProvider>(context);  final TextEditingController titleController = TextEditingController();  final TextEditingController descriptionController = TextEditingController();  return Scaffold(  appBar: AppBar(  title: const Text('Task Manager'),  ),  body: Column(  children: [  Padding(  padding: const EdgeInsets.all(16.0),  child: Column(  children: [  TextField(  controller: titleController,  decoration: const InputDecoration(  labelText: 'Task Title',  border: OutlineInputBorder(),  ),  ),  const SizedBox(height: 8.0),  TextField(  controller: descriptionController,  decoration: const InputDecoration(  labelText: 'Task Description',  border: OutlineInputBorder(),  ),  ),  const SizedBox(height: 8.0),  ElevatedButton(  onPressed: () {  if (titleController.text.isNotEmpty &&  descriptionController.text.isNotEmpty) {  taskProvider.addTask(  titleController.text,  descriptionController.text,  );  titleController.clear();  descriptionController.clear();  }  },  child: const Text('Add Task'),  ),  ],  ),  ),  Expanded(  child: Consumer<TaskProvider>(  builder: (context, taskProvider, child) {  return ListView.builder(  itemCount: taskProvider.tasks.length,  itemBuilder: (context, index) {  final task = taskProvider.tasks[index];  return ListTile(  leading: Checkbox(  value: task.isCompleted,  onChanged: (value) {  taskProvider.toggleTaskCompletion(index);  },  ),  title: Text(  task.title,  style: TextStyle(  decoration: task.isCompleted  ? TextDecoration.lineThrough  : TextDecoration.none,  ),  ),  subtitle: Text(task.description),  );  },  );  },  ),  ),  ],  ),  );  }  } |

OUTPUT:



**Task 2: Display User Data from API in a Flutter App**

**Objective:**Build a Flutter application that fetches and displays a list of users from a REST API using the http package and shows the user details in a ListView.

**Instructions:**

1. Use the following public API endpoint to retrieve user data:**https://jsonplaceholder.typicode.com/users**
2. Create a model class named User with the following fields:

* id (int)
* name (String)
* email (String)
* phone (String)

1. Implement a function to fetch user data from the API using http.get() and convert the JSON response into a list of User objects.
2. Use FutureBuilder to asynchronously display the list of users in a ListView.builder.
3. Each ListTile should display:

* **Title:** User’s name
* **Subtitle:** User’s email and phone number

1. Handle loading and error states properly.

Ans:

Main.dart:

|  |
| --- |
| import 'package:flutter/material.dart';  import 'package:http/http.dart' as http;  import 'dart:convert';  void main() {  runApp(const MyApp());  }    class MyApp extends StatelessWidget {  const MyApp({super.key});    @override  Widget build(BuildContext context) {  return MaterialApp(  title: 'User List Demo',  theme: ThemeData(  primarySwatch: Colors.blue,  visualDensity: VisualDensity.adaptivePlatformDensity,  ),  home: const UserListScreen(),  );  }  }    class User {  final int id;  final String name;  final String email;  final String phone;    User({  required this.id,  required this.name,  required this.email,  required this.phone,  });    factory User.fromJson(Map<String, dynamic> json) {  return User(  id: json['id'],  name: json['name'],  email: json['email'],  phone: json['phone'],  );  }  }    class UserListScreen extends StatefulWidget {  const UserListScreen({super.key});    @override  State<UserListScreen> createState() => \_UserListScreenState();  }    class \_UserListScreenState extends State<UserListScreen> {  Future<List<User>> fetchUsers() async {  final response = await http.get(  Uri.parse('https://jsonplaceholder.typicode.com/users'),  );    if (response.statusCode == 200) {  List<dynamic> data = jsonDecode(response.body);  return data.map((json) => User.fromJson(json)).toList();  } else {  throw Exception('Failed to load users');  }  }    @override  Widget build(BuildContext context) {  return Scaffold(  appBar: AppBar(  title: const Text('Users'),  ),  body: FutureBuilder<List<User>>(  future: fetchUsers(),  builder: (context, snapshot) {  if (snapshot.connectionState == ConnectionState.waiting) {  return const Center(child: CircularProgressIndicator());  } else if (snapshot.hasError) {  return Center(child: Text('Error: ${snapshot.error}'));  } else if (!snapshot.hasData || snapshot.data!.isEmpty) {  return const Center(child: Text('No users found'));  }    final users = snapshot.data!;  return ListView.builder(  itemCount: users.length,  itemBuilder: (context, index) {  final user = users[index];  return ListTile(  title: Text(user.name),  subtitle: Text('Email: ${user.email}\nPhone: ${user.phone}'),  );  },  );  },  ),  );  }  } |

OUTPUT:

