**СӨЖ**

**Жоба тақырыбы: Тапсырмалар тізімін басқару үшін To-Do қосымшасын әзірлеу**

Бұл код Flutter арқылы жазылған тапсырма менеджерін құруға арналған. Ол әртүрлі тапсырмаларды енгізуге, өңдеуге, белгілеуге, сақтау мен өшіруге мүмкіндік береді.

### **1. Негізгі құрылым (**main.dart**)**

**main() функциясы:**

* Flutter қосымшасын іске қосады.
* MyApp класын пайдаланып, қолданбаны бастайды.

**MyApp класы:**

* StatelessWidget-ке негізделген.
* Қосымшаның негізгі тақырыбын (theme) және басты бетін (HomePage) анықтайды.

### **2.** HomePage **класы**

**HomePage және \_HomePageState класы:**

* **StatefulWidget**: Бұл класс қолданбаның динамикалық функционалдылығын қамтамасыз етеді.

#### **Таңбашалар:**

* TextEditingController объектілері:
  + \_controllerTitle: Тапсырманың атауын сақтау үшін.
  + \_controllerDescription: Тапсырманың сипаттамасын сақтау үшін.
  + \_controllerDueDate: Тапсырманың мерзімін сақтау үшін.
* toDoList: Барлық тапсырмаларды сақтайтын тізім.

#### **SharedPreferences:**

* Тапсырмаларды сақтау мен жүктеу үшін пайдаланылады.
  + **saveTasks()**: toDoList-ті JSON пішімінде сақтайды.
  + **loadTasks()**: Сақталған тапсырмаларды жүктеп, toDoList тізіміне қайта жазады.

### **3. Негізгі функционалдылықтар**

#### **1. Тапсырма қосу (**saveNewTask**)**

* Жаңа тапсырма енгізілген мәліметтерді toDoList тізіміне қосады.
* Содан кейін SharedPreferences арқылы сақтайды.
* Енгізу өрістерін тазартады.

#### **2. Тапсырманы өңдеу (**editTask**)**

* Белгілі бір тапсырманың мәліметтерін редакциялау мүмкіндігін береді.
* Пайдаланушы AlertDialog арқылы тапсырманы өзгерте алады.

#### **3. Тапсырманы өшіру (**deleteTask**)**

* **Dismissible** виджеті арқылы тапсырманы оңай жоюға мүмкіндік береді.
* Жойылған тапсырма toDoList тізімінен өшіріледі.

#### **4. Тапсырманы аяқталды деп белгілеу (**checkBoxChanged**)**

* Тапсырманың мәртебесін (true/false) өзгертеді.

### **4. Интерфейс дизайны**

#### TodoList **класы:**

* Әрбір тапсырманы көрсетуге арналған арнайы виджет.
* **Құрамдас бөліктері:**
  + **Checkbox**: Тапсырманың орындалғанын белгілеу.
  + **IconButton**: Тапсырманы өңдеу үшін.
  + **Тапсырма деректері**:
    - Тақырып.
    - Сипаттама.
    - Мерзімі.

#### **Дизайн ерекшеліктері:**

* **Түстер**: Teal (көк-жасыл) түстер палитрасы.
* **Мәтін стилі**:
  + Мәтін аяқталған болса, үстінен сызылған сызықпен белгіленеді.
* **Тапсырма тізімі**: ListView.builder виджеті.

### **5. Қосымша мәліметтер**

#### **Тапсырмаларды сақтау:**

* SharedPreferences арқылы құрылғының жадында сақталады.
* Бұл әдіс қосымшаны қайта ашқанда тапсырмаларды жоғалтпауға мүмкіндік береді.

#### **Қателерді өңдеу:**

* Егер жүктелген мәлімет бос болса, toDoList автоматты түрде бастапқы бос күйінде қалады.

**ToDo-список жасауға арналған бағдарлама**  
Бұл бағдарлама күнді жоспарлауға және тапсырмаларды тиімді басқаруға көмектеседі. Оның көмегімен:

* **Тапсырмалар қосу**: Күнделікті жоспарларыңызды енгізе аласыз.
* **Тапсырмаларды өңдеу**: Жазылған мәліметтерді өзгертуге мүмкіндік бар.
* **Тапсырмаларды өшіру**: Қажетсіз тапсырмаларды оңай жоюға болады.
* **Орындалу мәртебесін белгілеу**: Тапсырмаларды орындалды немесе орындалмады деп белгілеу функциясы бар.
* **Уақытты енгізу**: Әр тапсырма үшін нақты орындалу мерзімін жазуға болады.
* **Деректерді сақтау**: Барлық енгізілген мәліметтер құрылғыда сақталады және қосымша жабылғаннан кейін де өшпейді (SharedPreferences арқылы).

Бұл бағдарлама күнделікті істеріңізді тиімді басқаруға көмектесетін қарапайым әрі ыңғайлы құрал болып табылады.

1. *Бұл кодты қолдана алу үшін сізде компьютеріңізде Flutter SDK орнатылған болуы керек. Егер орнатылмаған болса, ресми Flutter сайты арқылы орнатып алыңыз.*
2. *Одан кейін терминалда келесі команданы орындап, жаңа Flutter жобасын жасаңыз:*

*flutter create task\_manager  
cd task\_manager*

1. *Жоба қалтасының ішіндегі lib/main.dart файлын ашып, ішіндегі мазмұнды толығымен жойып, берілген кодты қойыңыз.*
2. *Кодта shared\_preferences пакеті қолданылған. Оны қосу үшін pubspec.yaml файлын ашып, келесі тәуелділікті енгізіңіз:*

*dependencies:  
flutter:  
sdk: flutter  
shared\_preferences: ^2.1.0*

1. *Терминалда келесі команданы орындап, бағдарламаны іске қосыңыз:*

*flutter run*

Бағдарлама коды:

import 'dart:convert';  
import 'package:flutter/material.dart';  
import 'package:shared\_preferences/shared\_preferences.dart';  
  
void main() {  
 runApp(const MyApp());  
}  
  
class MyApp extends StatelessWidget {  
 const MyApp({super.key});  
  
 @override  
 Widget build(BuildContext context) {  
 return MaterialApp(  
 debugShowCheckedModeBanner: false,  
 theme: ThemeData(  
 primarySwatch: Colors.teal,  
 visualDensity: VisualDensity.adaptivePlatformDensity,  
 textTheme: TextTheme(  
 bodyLarge: TextStyle(fontSize: 16, color: Colors.white),  
 bodyMedium: TextStyle(fontSize: 14, color: Colors.white70),  
 headlineLarge: TextStyle(fontSize: 24, fontWeight: FontWeight.bold, color: Colors.white),  
 ),  
 ),  
 home: HomePage(),  
 );  
 }  
}  
  
class HomePage extends StatefulWidget {  
 const HomePage({super.key});  
  
 @override  
 State<HomePage> createState() => \_HomePageState();  
}  
  
class \_HomePageState extends State<HomePage> {  
 final \_controllerTitle = TextEditingController();  
 final \_controllerDescription = TextEditingController();  
 final \_controllerDueDate = TextEditingController();  
 List toDoList = [];  
  
 @override  
 void initState() {  
 super.initState();  
 loadTasks();  
 }  
  
 Future<void> saveTasks() async {  
 SharedPreferences prefs = await SharedPreferences.getInstance();  
 String tasksJson = jsonEncode(toDoList);  
 await prefs.setString('toDoList', tasksJson);  
 }  
  
 Future<void> loadTasks() async {  
 SharedPreferences prefs = await SharedPreferences.getInstance();  
 String? savedTasks = prefs.getString('toDoList');  
  
 if (savedTasks != null) {  
 List<dynamic> tasks = jsonDecode(savedTasks);  
 setState(() {  
 toDoList = tasks;  
 });  
 }  
 }  
  
 void saveNewTask() {  
 setState(() {  
 toDoList.add([\_controllerTitle.text, \_controllerDescription.text, \_controllerDueDate.text, false]);  
 \_controllerTitle.clear();  
 \_controllerDescription.clear();  
 \_controllerDueDate.clear();  
 saveTasks();  
 });  
 }  
  
 void editTask(int index) {  
 \_controllerTitle.text = toDoList[index][0];  
 \_controllerDescription.text = toDoList[index][1];  
 \_controllerDueDate.text = toDoList[index][2];  
  
 showDialog(  
 context: context,  
 builder: (context) {  
 return AlertDialog(  
 backgroundColor: Colors.teal.shade800,  
 title: const Text('Edit Task', style: TextStyle(color: Colors.white)),  
 content: Column(  
 mainAxisSize: MainAxisSize.min,  
 children: [  
 TextField(  
 controller: \_controllerTitle,  
 decoration: const InputDecoration(  
 hintText: 'Task Title',  
 hintStyle: TextStyle(color: Colors.white60),  
 filled: true,  
 fillColor: Colors.teal,  
 border: OutlineInputBorder(),  
 ),  
 ),  
 TextField(  
 controller: \_controllerDescription,  
 decoration: const InputDecoration(  
 hintText: 'Task Description',  
 hintStyle: TextStyle(color: Colors.white60),  
 filled: true,  
 fillColor: Colors.teal,  
 border: OutlineInputBorder(),  
 ),  
 ),  
 TextField(  
 controller: \_controllerDueDate,  
 decoration: const InputDecoration(  
 hintText: 'Due Date',  
 hintStyle: TextStyle(color: Colors.white60),  
 filled: true,  
 fillColor: Colors.teal,  
 border: OutlineInputBorder(),  
 ),  
 ),  
 ],  
 ),  
 actions: [  
 TextButton(  
 onPressed: () {  
 setState(() {  
 toDoList[index][0] = \_controllerTitle.text;  
 toDoList[index][1] = \_controllerDescription.text;  
 toDoList[index][2] = \_controllerDueDate.text;  
 });  
 saveTasks();  
 \_controllerTitle.clear();  
 \_controllerDescription.clear();  
 \_controllerDueDate.clear();  
 Navigator.pop(context);  
 },  
 child: const Text('Save', style: TextStyle(color: Colors.white)),  
 ),  
 TextButton(  
 onPressed: () {  
 \_controllerTitle.clear();  
 \_controllerDescription.clear();  
 \_controllerDueDate.clear();  
 Navigator.pop(context);  
 },  
 child: const Text('Cancel', style: TextStyle(color: Colors.white)),  
 ),  
 ],  
 );  
 },  
 );  
 }  
  
 void checkBoxChanged(int index) {  
 setState(() {  
 toDoList[index][3] = !toDoList[index][3];  
 });  
 saveTasks();  
 }  
  
 void deleteTask(int index) {  
 setState(() {  
 toDoList.removeAt(index);  
 });  
 saveTasks();  
 }  
  
 @override  
 Widget build(BuildContext context) {  
 return Scaffold(  
 backgroundColor: Colors.teal.shade700,  
 appBar: AppBar(  
 title: const Text('Task Manager'),  
 backgroundColor: Colors.teal.shade900,  
 foregroundColor: Colors.white,  
 ),  
 body: ListView.builder(  
 itemCount: toDoList.length,  
 itemBuilder: (context, index) {  
 return Dismissible(  
 key: Key(toDoList[index][0]),  
 onDismissed: (direction) {  
 deleteTask(index);  
 },  
 background: Container(color: Colors.red),  
 child: FadeTransition(  
 opacity: AlwaysStoppedAnimation(1.0),  
 child: TodoList(  
 taskName: toDoList[index][0],  
 taskDescription: toDoList[index][1],  
 taskDueDate: toDoList[index][2],  
 taskCompleted: toDoList[index][3],  
 onChanged: (value) => checkBoxChanged(index),  
 onEdit: () => editTask(index),  
 ),  
 ),  
 );  
 },  
 ),  
 floatingActionButton: Padding(  
 padding: const EdgeInsets.symmetric(horizontal: 20),  
 child: Row(  
 children: [  
 Expanded(  
 child: Padding(  
 padding: const EdgeInsets.symmetric(horizontal: 20),  
 child: TextField(  
 controller: \_controllerTitle,  
 style: TextStyle(color: Colors.white),  
 decoration: InputDecoration(  
 hintText: 'Add a new task',  
 hintStyle: TextStyle(color: Colors.white60),  
 filled: true,  
 fillColor: Colors.teal.shade600,  
 enabledBorder: OutlineInputBorder(  
 borderSide: const BorderSide(color: Colors.teal),  
 borderRadius: BorderRadius.circular(10),  
 ),  
 focusedBorder: OutlineInputBorder(  
 borderSide: const BorderSide(color: Colors.teal),  
 borderRadius: BorderRadius.circular(10),  
 ),  
 ),  
 ),  
 ),  
 ),  
 FloatingActionButton(  
 onPressed: saveNewTask,  
 backgroundColor: Colors.orangeAccent,  
 child: const Icon(Icons.add),  
 ),  
 ],  
 ),  
 ),  
 );  
 }  
}  
  
class TodoList extends StatelessWidget {  
 const TodoList({  
 super.key,  
 required this.taskName,  
 required this.taskDescription,  
 required this.taskDueDate,  
 required this.taskCompleted,  
 required this.onChanged,  
 required this.onEdit,  
 });  
  
 final String taskName;  
 final String taskDescription;  
 final String taskDueDate;  
 final bool taskCompleted;  
 final Function(bool?)? onChanged;  
 final VoidCallback onEdit;  
  
 @override  
 Widget build(BuildContext context) {  
 return Padding(  
 padding: const EdgeInsets.only(top: 20, left: 20, right: 20, bottom: 0),  
 child: Container(  
 padding: const EdgeInsets.all(20),  
 decoration: BoxDecoration(  
 color: Colors.teal.shade800,  
 borderRadius: BorderRadius.circular(15),  
 ),  
 child: Row(  
 children: [  
 Checkbox(  
 value: taskCompleted,  
 onChanged: onChanged,  
 checkColor: Colors.black,  
 activeColor: Colors.white,  
 side: const BorderSide(color: Colors.white),  
 ),  
 Column(  
 crossAxisAlignment: CrossAxisAlignment.start,  
 children: [  
 Text(  
 taskName,  
 style: TextStyle(  
 color: Colors.white,  
 fontSize: 18,  
 decoration: taskCompleted ? TextDecoration.lineThrough : TextDecoration.none,  
 decorationColor: Colors.white,  
 decorationThickness: 2,  
 ),  
 ),  
 Text(  
 taskDescription,  
 style: TextStyle(color: Colors.white),  
 ),  
 Text(  
 'Due: $taskDueDate',  
 style: TextStyle(color: Colors.white70),  
 ),  
 ],  
 ),  
 IconButton(  
 icon: const Icon(Icons.edit, color: Colors.white),  
 onPressed: onEdit,  
 ),  
 ],  
 ),  
 ),  
 );  
 }  
}

Бағдарлама нәтижесі интерфейсі:

