**СӨЖ**

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

**Орындалу кезеңдері мен мерзімі:**

**1. Кезең 1: Интерфейсті жобалау және макет құру**

* Flutter негіздерін үйрену және басты экран мен тапсырма қосу экраны үшін пайдаланушы интерфейсі әзірленді.
* Визуалды макет құру үшін Flutter-дің кіріктірілген виджеттерін пайдаланылды.
* **Нәтиже**: Интерфейстің макеті бар басты экран мен тапсырма қосу экранының жобасы.

**2. Кезең 2: Тапсырмаларды қосу және өңдеу функцияларын іске асыру**

* Жаңа тапсырмаларды атауы, сипаттамасы және аяқталу күнін қосу мүмкіндігі жасалды.
* Бар тапсырмаларды өңдеу функциясы қосылды.
* Деректерді енгізу оқиғаларын өңдеу және қосымшада көрсету.
* **Нәтиже**: Жаңа тапсырмаларды қосу және өңдеу мүмкіндігі бар интерфейс.

**3. Кезең 3: Тапсырмаларды жою және күйін өзгерту**

* Тапсырмаларды сырғыту арқылы жою функциясы қосылды.
* Тапсырманың күйін өзгерту (орындалды/орындалмады) және тізімде көрсетілді.
* **Нәтиже**: Тапсырмалардың күйін өзгерту және оларды жою мүмкіндігі.

**4. Кезең 4: Деректерді сақтау және жергілікті сақтау қоймасын орнату**

* Тапсырмаларды сақтау үшін жергілікті сақтау механизмі (SharedPreferences) қосылды.
* Қосымша қайта ашылғанда, тапсырмалардың сақталуын қамтамасыз ету.
* **Нәтиже**: Қосымшаның сессиялар арасында деректерді сақтай алуы.

**5. Кезең 5: Тапсырмаларды сүзгілеу және соңғы тексеру**

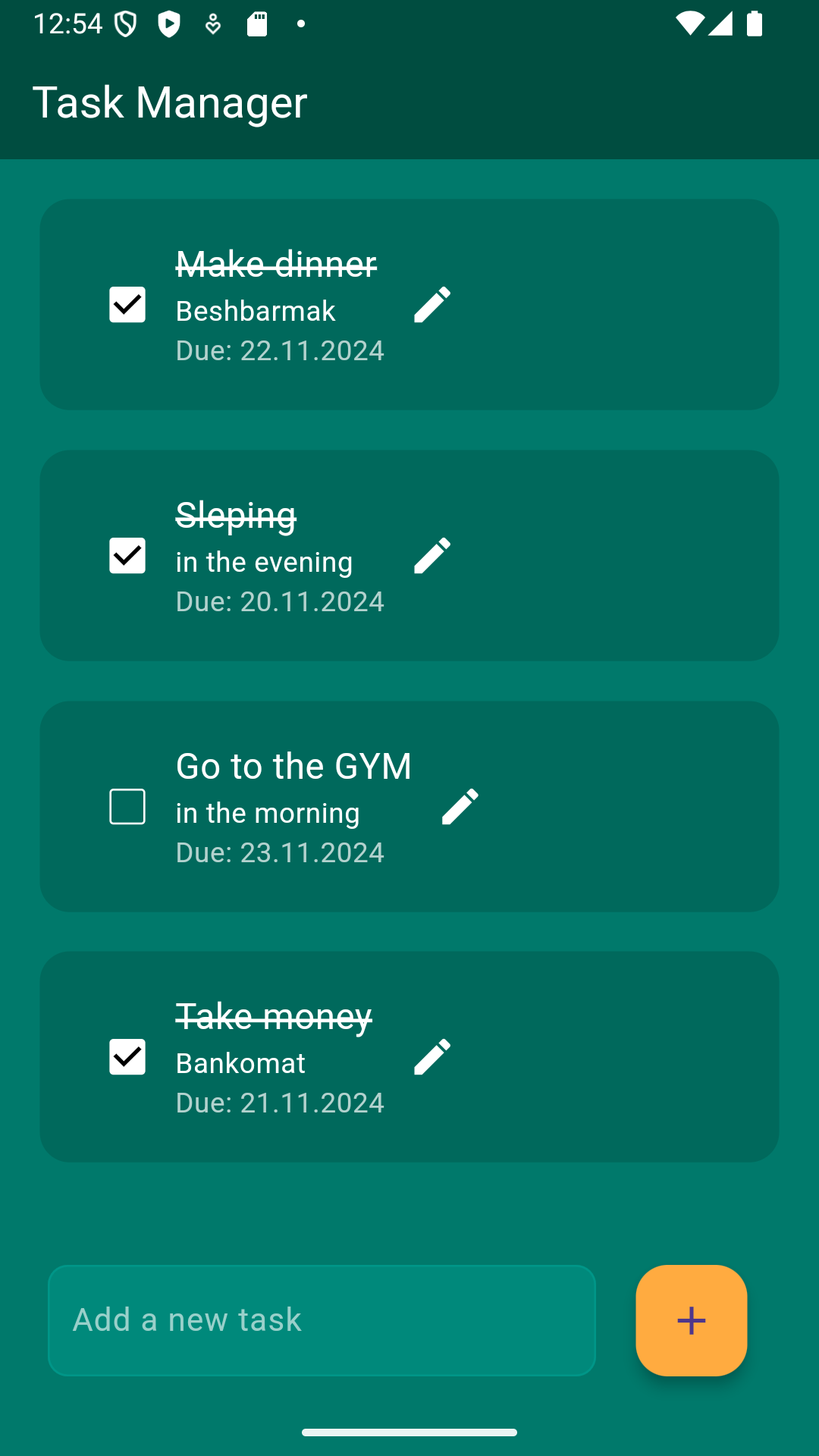
* Тапсырмаларды күйі бойынша сүзгілеу функциясын қосу (барлық, орындалған, орындалмаған).
* Қосымшаның барлық функцияларын тексеру және қателерін жою.
* Интерфейсті және пайдаланушы тәжірибесін жақсарту үшін соңғы түзетулер жасалды.

**Нәтиже**: Тапсырмаларды сүзгілеу және деректерді сақтау функциялары бар толық жұмыс істейтін қосымша.

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:



Нәтижесінде ToDo-список задач бағдарламасы жасалынды. Бұл бағдарлама арқылы күнді жоспарлауға, тапсырмалар жазуға болады.Жазылған мәліметтерді өшіруге және өңдеуге болады. Тапсырмаларды орындалды немесе орындалмады деп белгілеуге болады. Сонымен қатар уақытын да жазса болады. Жазылған мәліметтер сақталынып отырады және өшіп кетпейді(SharedPreferences арқылы).