PRACTICAL 4

Aim: To create an interactive Form using form widget .

Theory:
The `Form` widget in Flutter is a fundamental building block for creating forms and handling user input. In the code provided, the `Form` widget is used to manage the state and validation of the input fields. Here's a detailed breakdown of how it works in this context:

Why Use the Form Widget**?**

- **State Management**: It simplifies managing multiple input fields by grouping them under one state.
- **Validation**: Provides a structured way to validate user inputs.
- **Scalability**: You can add more fields easily without affecting the overall structure.
- **Code Organization**: Keeps validation logic separate from the rest of the UI code.

CODE
Folder Structure

```
lib/
├── screens/
│   ├── home_screen.dart
│   ├── login_screen.dart
│   ├── features/
│   │   ├── medication_reminders_screen.dart
│   │   ├── add_medication_reminder_screen.dart
```

medication_reminders_screen.dart:

```
import 'package:flutter/material.dart';
import 'add_medication_reminder_screen.dart';

class MedicationRemindersScreen extends StatefulWidget {
  const MedicationRemindersScreen({super.key});

  @override
  State<MedicationRemindersScreen> createState() => _MedicationRemindersScreenState();
}

class _MedicationRemindersScreenState extends State<MedicationRemindersScreen> {
  @override
```

```
  Widget build(BuildContext context) {
    return Center(
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
          Icon(
            Icons.medication,
            size: 80,
            color: Colors.blue.shade700,
          ),
          const SizedBox(height: 16),
          const Text(
            'Medication Reminders',
            style: TextStyle(
              fontSize: 24,
              fontWeight: FontWeight.bold,
            ),
          ),
          const SizedBox(height: 24),
          ElevatedButton.icon(
            style: ElevatedButton.styleFrom(
              backgroundColor: Colors.blue.shade700,
              foregroundColor: Colors.white,
              padding: const EdgeInsets.symmetric(horizontal: 24, vertical: 12),
              shape: RoundedRectangleBorder(
                borderRadius: BorderRadius.circular(30),
              ),
            ),
            onPressed: () {
              Navigator.push(
                context,
                MaterialPageRoute(builder: (context) => const AddMedicationReminderScreen()),
              );
            },
            icon: const Icon(Icons.add),
            label: const Text('Add Medication Reminder', style: TextStyle(fontSize: 16)),
          ),
        ],
      ),
    );
  }
}
```

add_medication_reminder_screen.dart

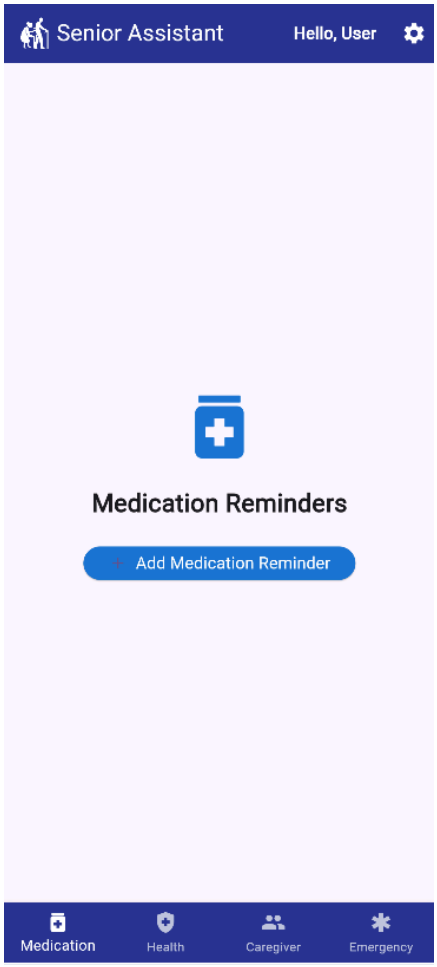
```
import 'package:flutter/material.dart';

class AddMedicationReminderScreen extends StatefulWidget {
  const AddMedicationReminderScreen({super.key});

  @override
  State<AddMedicationReminderScreen> createState() =>
_AddMedicationReminderScreenState();
}

class _AddMedicationReminderScreenState extends State<AddMedicationReminderScreen> {
  final _formKey = GlobalKey<FormState>();
  final TextEditingController _medicineNameController = TextEditingController();
  final TextEditingController _dosageController = TextEditingController();
  final TextEditingController _timeController = TextEditingController();
  TimeOfDay? _selectedTime;

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Add Medication Reminder'),
      ),
      body: Padding(
        padding: const EdgeInsets.all(16.0),
        child: Form(
          key: _formKey,
          child: ListView(
            children: [
              // Medicine Name Field
              TextFormField(
                controller: _medicineNameController,
                decoration: const InputDecoration(
                  labelText: 'Medicine Name',
                  border: OutlineInputBorder(),
                ),
                validator: (value) {
                  if (value == null || value.isEmpty) {
                    return 'Please enter the medicine name';
                  }
                  return null;
                },
              ),
```

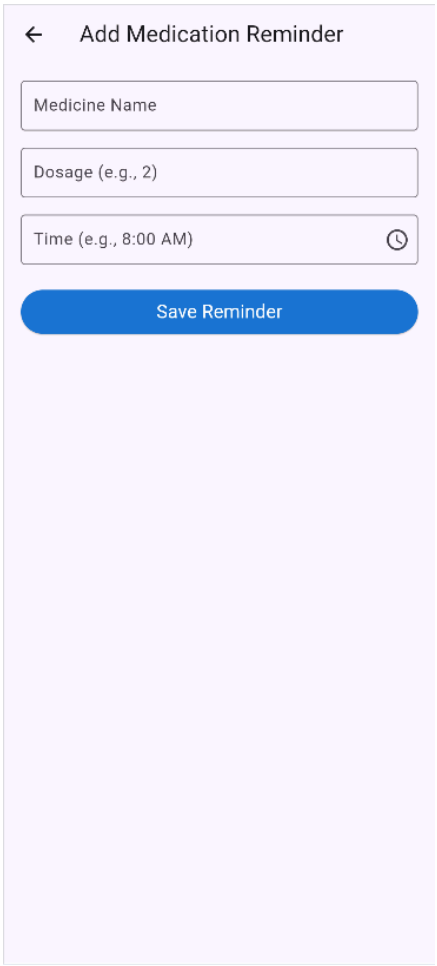
```
const SizedBox(height: 16),

// Dosage Field (Number Input)
TextFormField(
  controller: _dosageController,
  keyboardType: TextInputType.number,
  decoration: const InputDecoration(
    labelText: 'Dosage (e.g., 2)',
    border: OutlineInputBorder(),
  ),
  validator: (value) {
    if (value == null || value.isEmpty) {
      return 'Please enter the dosage';
    }
    if (int.tryParse(value) == null) {
      return 'Please enter a valid number';
    }
    return null;
  },
),
const SizedBox(height: 16),

// Time Field (Time Picker)
TextFormField(
  controller: _timeController,
  readOnly: true,
  decoration: const InputDecoration(
    labelText: 'Time (e.g., 8:00 AM)',
    border: OutlineInputBorder(),
    suffixIcon: Icon(Icons.access_time),
  ),
  onTap: () async {
    // Show Time Picker
    TimeOfDay? pickedTime = await showTimePicker(
      context: context,
      initialTime: TimeOfDay.now(),
    );

    if (pickedTime != null) {
      setState(() {
        _selectedTime = pickedTime;
        _timeController.text = pickedTime.format(context);
      });
    }
```

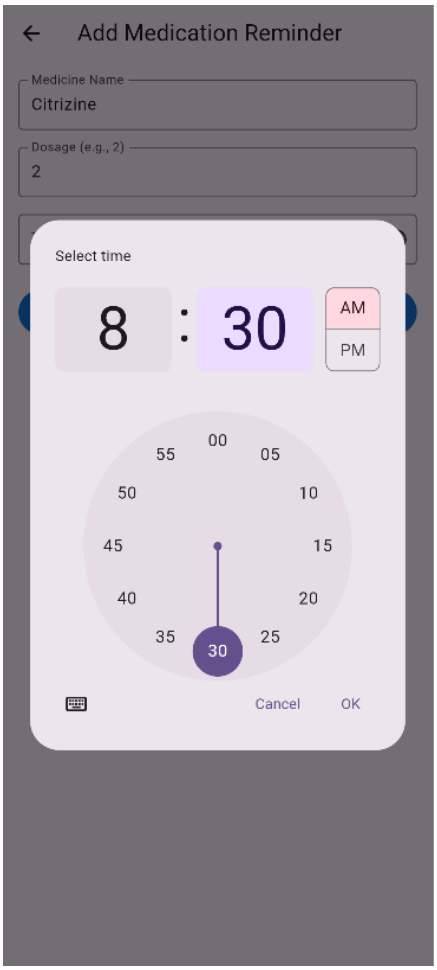
```
            },
            validator: (value) {
              if (value == null || value.isEmpty) {
                return 'Please select a time';
              }
              return null;
            },
          ),
          const SizedBox(height: 24),

          // Save Reminder Button
          ElevatedButton(
            style: ElevatedButton.styleFrom(
              backgroundColor: Colors.blue.shade700,
              foregroundColor: Colors.white,
              padding: const EdgeInsets.symmetric(vertical: 16),
            ),
            onPressed: () {
              if (_formKey.currentState!.validate()) {
                // Process the input data
                String medicineName = _medicineNameController.text;
                int dosage = int.parse(_dosageController.text);
                String time = _timeController.text;

                // Show confirmation message
                ScaffoldMessenger.of(context).showSnackBar(
                  SnackBar(
                    content: Text(
                      'Medication Reminder Added:\nMedicine: $medicineName\nDosage:
$dosage\nTime: $time',
                    ),
                  ),
                );

                // Clear form after submission
                _medicineNameController.clear();
                _dosageController.clear();
                _timeController.clear();
              }
            },
            child: const Text('Save Reminder', style: TextStyle(fontSize: 18)),
          ),
        ],
      ),
```

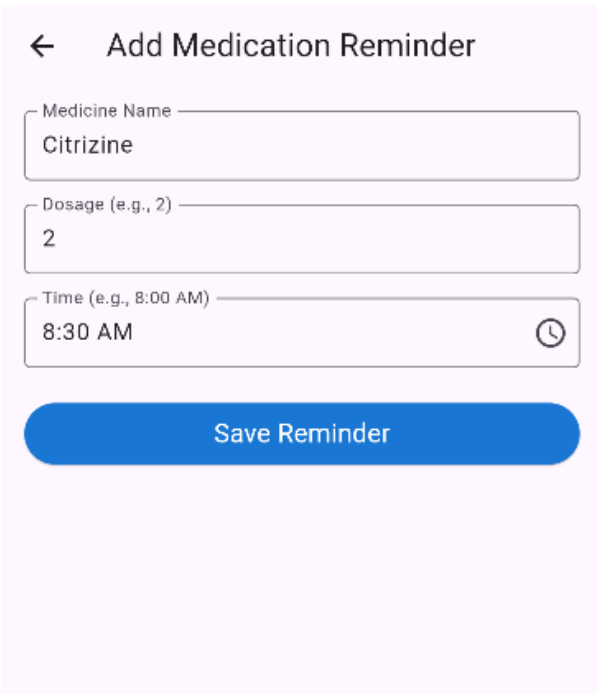
```
        ),
      ),
    );
  }
}
```

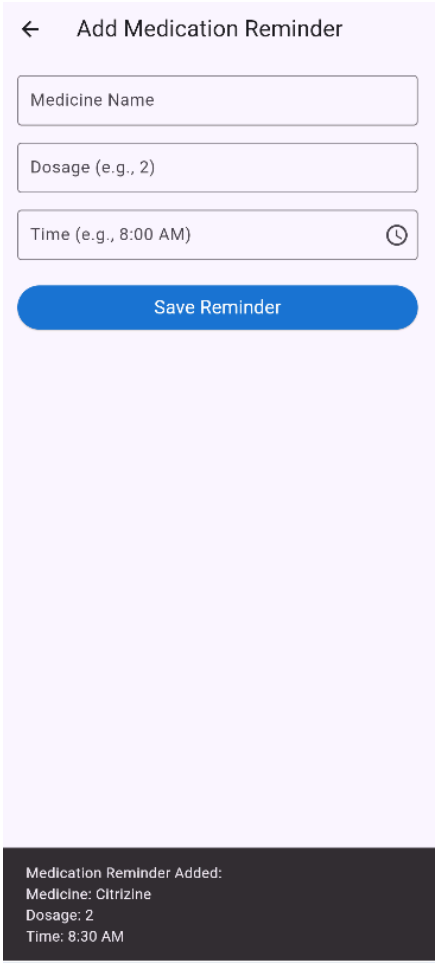
Screenshot:


Senior Assistant
Hello, User
Medication Reminders
Add Medication Reminder
Medication
Health
Caregiver
Emergency


Add Medication Reminder
Medicine Name
Dosage (e.g., 2)
Time (e.g., 8:00 AM)
Save Reminder


Add Medication Reminder
Medicine Name
Citrizine
Dosage (e.g., 2)
2
Select time
8
:
30
AM
PM
00
05
10
15
20
25
30
35
40
45
50
55
Cancel
OK


Add Medication Reminder
Medicine Name
Citrizine
Dosage (e.g., 2)
2
Time (e.g., 8:00 AM)
8:30 AM
Save Reminder



Conclusion

During the implementation, I encountered errors like incorrect type conversions (e.g., parsing `String` to `int` for dosage) and validation failures when fields were left empty. These issues were resolved by adding robust validation logic for each field and displaying clear error messages. Proper type handling and clearing the controllers after submission ensured smooth functionality and improved user experience.