PRACTICAL: 1

AIM:

You are building a mobile application where users navigate through multiple screens like login, dashboard, and profile. Create a basic multi-screen Flutter app with navigation, passing data between pages.

THEORY:

Navigation in Flutter refers to moving between different screens in an app. Routing is the mechanism that defines how navigation happens mapping route names to the widgets they represent.

CODE:

```
// login screen

import 'package:flutter/material.dart';


class LoginScreen extends StatefulWidget {

  const LoginScreen({super.key});


  @override

  State<LoginScreen> createState() => _LoginScreenState();

}


class _LoginScreenState extends State<LoginScreen> {

  final TextEditingController _nameController = TextEditingController();


  void _login() {

    String name = _nameController.text.trim();


    if (name.isNotEmpty) {

      Navigator.pushReplacementNamed(context, '/dashboard', arguments: name);

    } else {
```

```
    ScaffoldMessenger.of(
      context,
    ).showSnackBar(SnackBar(content: Text('Please enter your name')));
  }
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    backgroundColor: Colors.deepPurple[50],
    appBar: AppBar(
      title: Text('Login'),
      backgroundColor: const Color.fromARGB(255, 250, 249, 251),
    ),
    body: Padding(
      padding: const EdgeInsets.all(24),
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
          Icon(Icons.lock, size: 60, color: Colors.deepPurple),
          SizedBox(height: 20),
          TextField(
            controller: _nameController,
            decoration: InputDecoration(
              labelText: 'Enter your name',
              border: OutlineInputBorder(),
            ),
```

```
            ),
            SizedBox(height: 20),
            ElevatedButton.icon(
              onPressed: _login,
              icon: Icon(Icons.login),
              label: Text('Login'),
              style: ElevatedButton.styleFrom(
                backgroundColor: const Color.fromARGB(255, 250, 250, 250),
                minimumSize: Size.fromHeight(50),
              ),
            ),
          ],
        ),
      ),
    );
  }
}

//dashboard page
import 'package:flutter/material.dart';
import 'profile_page.dart';

class DashboardScreen extends StatelessWidget {
  const DashboardScreen({super.key});

  @override
  Widget build(BuildContext context) {
```

```
final String userName =
    ModalRoute.of(context)!.settings.arguments as String;


return Scaffold(
  backgroundColor: Colors.white,
  appBar: AppBar(
    title: Text('Dashboard'),
    backgroundColor: const Color.fromARGB(255, 253, 253, 254),
    actions: [
      IconButton(
        icon: Icon(Icons.logout),
        onPressed: () {
          Navigator.pushReplacementNamed(context, '/login');
        },
      ),
    ],
  ),
  body: Center(
    child: Column(
      mainAxisAlignment: MainAxisAlignment.center,
      children: [
        Icon(Icons.dashboard, size: 100, color: Colors.deepPurple),
        SizedBox(height: 20),
        Text(
          'Welcome, $userName!',
          style: TextStyle(fontSize: 26, fontWeight: FontWeight.bold),
        ),
```

```
          SizedBox(height: 10),
          Text(
            'You are now logged in to the demo app.',
            style: TextStyle(fontSize: 16, color: Colors.grey[600]),
          ),
          SizedBox(height: 30),
          ElevatedButton.icon(
            icon: Icon(Icons.person),
            label: Text('Go to Profile'),
            onPressed: () {
              Navigator.push(
                context,
                MaterialPageRoute(
                  builder: (context) => ProfilePage(userName: userName),
                ),
              );
            },
          ),
        ],
      ),
    ),
  );
 }
}


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

```
class ProfilePage extends StatelessWidget {
  final String userName;
  const ProfilePage({super.key, required this.userName});

  @override
  Widget build(BuildContext context) {
   return Scaffold(
     appBar: AppBar(
       title: const Text('Profile'),
       leading: IconButton(
         icon: const Icon(Icons.arrow_back),
         onPressed: () => Navigator.of(context).pop(),
       ),
     ),
     body: Center(
       child: Column(
         mainAxisAlignment: MainAxisAlignment.center,
         children: [
           const CircleAvatar(radius: 40, child: Icon(Icons.person, size: 50)),
           const SizedBox(height: 20),
           Text(
             userName,
             style: const TextStyle(fontSize: 24, fontWeight: FontWeight.bold),
           ),
           const SizedBox(height: 10),
           const Text('user@email.com', style: TextStyle(fontSize: 16)),
```

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

OUTPUT:

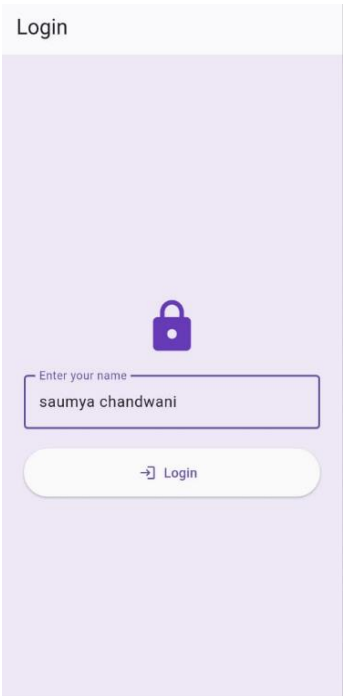


Fig. 1. Login Fig. 2. Dashboard Fig. 3. Profile

In the screenshots we see the ui of the application, in the first page when entering data in the controllers and then pressing login, it takes us to the dashboard, then pressing the profile icon in dashboard appbar takes us to the profile page where the name and email from the login screen are reflected

LATEST APPLICATIONS:

A basic multi-screen Flutter app with navigation and data passing is widely used in modern e-commerce and social media apps to manage user flows like login, dashboards, and profiles. It's also

applied in education and productivity apps to provide seamless multi-screen experiences with personalized data.

LEARNING OUTCOME:

By completing this practical 1 learnt how to implement navigation and routing between pages using ongenerate, routes and navigator functions like push, replacement etc.

REFERENCES:

1. **Flutter Docs – Navigation and Routing** – https://docs.flutter.dev/ui/navigation
2. **Flutter Cookbook: Navigate to a new screen and back** – https://docs.flutter.dev/cookbook/navigation/navigation-basics
3. **Flutter Cookbook: Send data to a new screen** – https://docs.flutter.dev/cookbook/navigation/passing-data

PRACTICAL: 2

AIM:

You are building a mobile application where users navigate through multiple screens like login, dashboard, and profile. Develop a temperature converter app using Dart functions and input widgets.

THEORY:

Use of stateful widgets is done in pages or elements where data is expected to change after the widget is built, it can be a calculator result element, which doesn't have to be a constant value as it keeps changing on new calculations.

CODE:

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

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

  @override
  _TemperatureConverterPageState createState() =>
      _TemperatureConverterPageState();
}

class _TemperatureConverterPageState extends State<TemperatureConverterPage> {
  final _inputController = TextEditingController();
  String _fromUnit = 'Celsius';
  String _toUnit = 'Fahrenheit';
  String _result = '';
  final List<String> _conversionHistory = [];

  final List<String> _units = ['Celsius', 'Fahrenheit', 'Kelvin'];

  // Temperature conversion functions
  double celsiusToFahrenheit(double celsius) {
    return (celsius * 9 / 5) + 32;
  }

  double celsiusToKelvin(double celsius) {
    return celsius + 273.15;
  }

  double fahrenheitToCelsius(double fahrenheit) {
```

```
  return (fahrenheit - 32) * 5 / 9;
 }

 double fahrenheitToKelvin(double fahrenheit) {
  return celsiusToKelvin(fahrenheitToCelsius(fahrenheit));
 }

 double kelvinToCelsius(double kelvin) {
  return kelvin - 273.15;
 }

 double kelvinToFahrenheit(double kelvin) {
  return celsiusToFahrenheit(kelvinToCelsius(kelvin));
 }

 double convertTemperature(double value, String from, String to) {
  if (from == to) return value;

  switch (from) {
    case 'Celsius':
      if (to == 'Fahrenheit') return celsiusToFahrenheit(value);
      if (to == 'Kelvin') return celsiusToKelvin(value);
      break;
    case 'Fahrenheit':
      if (to == 'Celsius') return fahrenheitToCelsius(value);
      if (to == 'Kelvin') return fahrenheitToKelvin(value);
      break;
    case 'Kelvin':
      if (to == 'Celsius') return kelvinToCelsius(value);
      if (to == 'Fahrenheit') return kelvinToFahrenheit(value);
      break;
  }
  return value;
 }

 void _convert() {
  if (_inputController.text.isEmpty) {
    setState(() {
      _result = 'Please enter a temperature value';
    });
    return;
  }

  try {
    double inputValue = double.parse(_inputController.text);
```

```
      double convertedValue = convertTemperature(
        inputValue,
        _fromUnit,
        _toUnit,
      );

      setState(() {
        _result = '${convertedValue.toStringAsFixed(2)}° $_toUnit';
        _conversionHistory.insert(
          0,
          '${inputValue.toStringAsFixed(2)}° $_fromUnit → ${convertedValue.toStringAsFixed(2)}°
$_toUnit',
        );

        // Keep only last 10 conversions
        if (_conversionHistory.length > 10) {
          _conversionHistory.removeRange(10, _conversionHistory.length);
        }
      });
    } catch (e) {
      setState(() {
        _result = 'Invalid input. Please enter a valid number.';
      });
    }
  }

  void _clear() {
    setState(() {
      _inputController.clear();
      _result = '';
    });
  }

  void _clearHistory() {
    setState(() {
      _conversionHistory.clear();
    });
  }

  void _swapUnits() {
    setState(() {
      String temp = _fromUnit;
      _fromUnit = _toUnit;
      _toUnit = temp;
    });
```

```
  if (_inputController.text.isNotEmpty) {
    _convert();
  }
 }

 @override
 Widget build(BuildContext context) {
  return Scaffold(
   appBar: AppBar(
    title: const Text('Temperature Converter'),
    backgroundColor: Colors.orange,
    elevation: 0,
    actions: [
     IconButton(
      icon: const Icon(Icons.swap_horiz),
      onPressed: _swapUnits,
      tooltip: 'Swap Units',
     ),
    ],
   ),
   body: Container(
    decoration: BoxDecoration(
     gradient: LinearGradient(
      begin: Alignment.topCenter,
      end: Alignment.bottomCenter,
      colors: [Colors.orange.shade100, Colors.white],
     ),
    ),
    child: SafeArea(
     child: SingleChildScrollView(
      padding: const EdgeInsets.all(20.0),
      child: Column(
       children: [
        // Input Section
        Card(
         elevation: 5,
         shape: RoundedRectangleBorder(
          borderRadius: BorderRadius.circular(15),
         ),
         child: Padding(
          padding: const EdgeInsets.all(20.0),
          child: Column(
           children: [
            TextField(
             controller: _inputController,
```

```
  keyboardType: const TextInputType.numberWithOptions(
   decimal: true,
  ),
  decoration: InputDecoration(
   labelText: 'Enter temperature value',
   prefixIcon: const Icon(Icons.thermostat),
   border: OutlineInputBorder(
    borderRadius: BorderRadius.circular(10),
   ),
  ),
 ),
 const SizedBox(height: 20),
 LayoutBuilder(
  builder: (context, constraints) {
   // If width is less than 400, use Column instead of Row
   if (constraints.maxWidth < 400) {
    return Column(
     children: [
      DropdownButtonFormField<String>(
       value: _fromUnit,
       decoration: InputDecoration(
        labelText: 'From',
        border: OutlineInputBorder(
         borderRadius: BorderRadius.circular(10),
        ),
       ),
       items: _units.map((String unit) {
        return DropdownMenuItem<String>(
         value: unit,
         child: Text(unit),
        );
       }).toList(),
       onChanged: (String? newValue) {
        setState(() {
         _fromUnit = newValue!;
        });
        if (_inputController.text.isNotEmpty) {
         _convert();
        }
       },
      ),
      const SizedBox(height: 20),
      DropdownButtonFormField<String>(
       value: _toUnit,
       decoration: InputDecoration(
```

```
                labelText: 'To',
                border: OutlineInputBorder(
                  borderRadius: BorderRadius.circular(10),
                ),
              ),
              items: _units.map((String unit) {
                return DropdownMenuItem<String>(
                  value: unit,
                  child: Text(unit),
                );
              }).toList(),
              onChanged: (String? newValue) {
                setState(() {
                  _toUnit = newValue!;
                });
                if (_inputController.text.isNotEmpty) {
                  _convert();
                }
              },
            ),
          ],
        );
      } else {
        return Row(
          children: [
            Expanded(
              child: DropdownButtonFormField<String>(
                value: _fromUnit,
                decoration: InputDecoration(
                  labelText: 'From',
                  border: OutlineInputBorder(
                    borderRadius: BorderRadius.circular(10),
                  ),
                ),
                items: _units.map((String unit) {
                  return DropdownMenuItem<String>(
                    value: unit,
                    child: Text(unit),
                  );
                }).toList(),
                onChanged: (String? newValue) {
                  setState(() {
                    _fromUnit = newValue!;
                  });
                  if (_inputController.text.isNotEmpty) {
```

```
                    _convert();
                  }
                },
              ),
            ),
            const SizedBox(width: 20),
            Expanded(
              child: DropdownButtonFormField<String>(
                value: _toUnit,
                decoration: InputDecoration(
                  labelText: 'To',
                  border: OutlineInputBorder(
                    borderRadius: BorderRadius.circular(10),
                  ),
                ),
                items: _units.map((String unit) {
                  return DropdownMenuItem<String>(
                    value: unit,
                    child: Text(unit),
                  );
                }).toList(),
                onChanged: (String? newValue) {
                  setState(() {
                    _toUnit = newValue!;
                  });
                  if (_inputController.text.isNotEmpty) {
                    _convert();
                  }
                },
              ),
            ),
          ],
        );
      }
    },
  ),
  const SizedBox(height: 20),
  Row(
    children: [
      Expanded(
        child: ElevatedButton(
          onPressed: _convert,
          style: ElevatedButton.styleFrom(
            backgroundColor: Colors.orange,
            padding: const EdgeInsets.symmetric(vertical: 15),
```

```
                    shape: RoundedRectangleBorder(
                      borderRadius: BorderRadius.circular(10),
                    ),
                  ),
                  child: const Text(
                    'Convert',
                    style: TextStyle(
                      fontSize: 16,
                      color: Colors.white,
                    ),
                  ),
                ),
              ),
              const SizedBox(width: 10),
              Expanded(
                child: ElevatedButton(
                  onPressed: _clear,
                  style: ElevatedButton.styleFrom(
                    backgroundColor: Colors.grey,
                    padding: const EdgeInsets.symmetric(vertical: 15),
                    shape: RoundedRectangleBorder(
                      borderRadius: BorderRadius.circular(10),
                    ),
                  ),
                  child: const Text(
                    'Clear',
                    style: TextStyle(
                      fontSize: 16,
                      color: Colors.white,
                    ),
                  ),
                ),
              ),
            ],
          ),
        ],
      ),
    ),
  ),
  const SizedBox(height: 20),
  // Result Section
  if (_result.isNotEmpty)
    Card(
      elevation: 5,
      color: Colors.orange.shade50,
```

```
          shape: RoundedRectangleBorder(
            borderRadius: BorderRadius.circular(15),
          ),
          child: Padding(
            padding: const EdgeInsets.all(20.0),
            child: Column(
              children: [
                const Icon(
                  Icons.thermostat,
                  size: 50,
                  color: Colors.orange,
                ),
                const SizedBox(height: 10),
                const Text(
                  'Result:',
                  style: TextStyle(
                    fontSize: 18,
                    fontWeight: FontWeight.bold,
                  ),
                ),
                const SizedBox(height: 10),
                Text(
                  _result,
                  style: TextStyle(
                    fontSize: 24,
                    fontWeight: FontWeight.bold,
                    color: Colors.orange.shade800,
                  ),
                ),
              ],
            ),
          ),
        ),
      const SizedBox(height: 20),
      // History Section
      if (_conversionHistory.isNotEmpty)
        Card(
          elevation: 5,
          shape: RoundedRectangleBorder(
            borderRadius: BorderRadius.circular(15),
          ),
          child: Padding(
            padding: const EdgeInsets.all(20.0),
            child: Column(
              crossAxisAlignment: CrossAxisAlignment.start,
```

```
                children: [
                  Row(
                    mainAxisAlignment: MainAxisAlignment.spaceBetween,
                    children: [
                      const Text(
                        'Conversion History',
                        style: TextStyle(
                          fontSize: 18,
                          fontWeight: FontWeight.bold,
                        ),
                      ),
                      IconButton(
                        icon: const Icon(Icons.clear_all),
                        onPressed: _clearHistory,
                        tooltip: 'Clear History',
                      ),
                    ],
                  ),
                  const SizedBox(height: 10),
                  ListView.builder(
                    shrinkWrap: true,
                    physics: const NeverScrollableScrollPhysics(),
                    itemCount: _conversionHistory.length,
                    itemBuilder: (context, index) {
                      return Padding(
                        padding: const EdgeInsets.symmetric(vertical: 5),
                        child: Text(
                          _conversionHistory[index],
                          style: TextStyle(
                            fontSize: 14,
                            color: Colors.grey.shade700,
                          ),
                        ),
                      );
                    },
                  ),
                ],
              ),
            ),
          ),
        ],
      ),
    ),
```

```
    );
  }

  @override
  void dispose() {
    _inputController.dispose();
    super.dispose();
  }
}
```

OUTPUT:

Fig. 1. Celsius to Fahrenheit

LATEST APPLICATIONS:

Use of flutter stateful and stateless widgets in the correct places can help manage resources occupied by the application as flutter allows dynamic as well as static pages

LEARNING OUTCOME:

Learning real world usage of stateless stateful widgets for building production ready applications

REFERENCES:

- **Flutter Docs** – Material Components (TextField, Button) – https://docs.flutter.dev/development/ui/widgets/material
- **Flutter Community Medium – Flutter Input Widgets and State Management** – https://medium.com/flutter-community/flutter-input-widgets-and-state-management-80a129ca06ec
- **YouTube – Flutter Beginner Tutorial: Temperature Converter App** – https://www.youtube.com/watch?v=J9WqnjLP9Z8

PRACTICAL: 3

AIM:

You are building a mobile application where users navigate through multiple screens like login, dashboard, and profile. Create a dynamic TODO app using State Management (setState) and ListView.builder.

THEORY:

setState is a method used in Flutter's StatefulWidgets to update the UI when data changes.

When call setState is called, Flutter rebuilds the widget tree for that widget, reflecting any changes in variables.

ListView is a scrollable list of widgets in Flutter.

ListView.builder is a constructor that builds list items on demand, which is efficient for long or dynamic lists.

CODE:

```
// to do page

import 'package:flutter/material.dart';

import '../models/todo_item.dart';

import 'profile_screen.dart';


class DashboardScreen extends StatefulWidget {

  final String userEmail;


  const DashboardScreen({super.key, required this.userEmail});


  @override

  State<DashboardScreen> createState() => _DashboardScreenState();
```

```
}

class _DashboardScreenState extends State<DashboardScreen> {
  final List<TodoItem> _todoItems = [];
  final _titleController = TextEditingController();
  final _descriptionController = TextEditingController();
  String _selectedCategory = 'Personal';
  String _filterCategory = 'All';
  bool _showCompleted = true;

  final List<String> _categories = [
    'Personal',
    'Work',
    'Shopping',
    'Health',
    'Education',
  ];

  @override
  void initState() {
    super.initState();
    // No sample items - start with empty list
  }

  void _addTodoItem() {
    if (_titleController.text.isNotEmpty) {
      setState(() {
```

```
      _todoItems.add(
        TodoItem(
          id: DateTime.now().millisecondsSinceEpoch.toString(),
          title: _titleController.text,
          description: _descriptionController.text,
          category: _selectedCategory,
          createdAt: DateTime.now(),
        ),
      );
    });
    _titleController.clear();
    _descriptionController.clear();
    Navigator.pop(context);
  }
}

void _deleteTodoItem(String id) {
  setState(() {
    _todoItems.removeWhere((item) => item.id == id);
  });
}

void _toggleTodoItem(String id) {
  setState(() {
    final item = _todoItems.firstWhere((item) => item.id == id);
    item.isCompleted = !item.isCompleted;
  });
```

```
}

List<TodoItem> get _filteredItems {
  return _todoItems.where((item) {
    bool categoryMatch =
        _filterCategory == 'All' || item.category == _filterCategory;
    bool completionMatch = _showCompleted || !item.isCompleted;
    return categoryMatch && completionMatch;
  }).toList();
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: const Text('TODO Dashboard'),
      backgroundColor: Theme.of(context).colorScheme.inversePrimary,
      actions: [
        IconButton(
          icon: const Icon(Icons.person),
          onPressed: () {
            Navigator.push(
              context,
              MaterialPageRoute(
                builder: (context) =>
                    ProfileScreen(userEmail: widget.userEmail),
              ),
```

```
          );
        },
      ),
    ],
  ),
  body: Column(
    children: [
      // Filter Section
      Container(
        padding: const EdgeInsets.all(16),
        child: Column(
          children: [
            // Category Filter
            DropdownButtonFormField<String>(
              value: _filterCategory,
              decoration: const InputDecoration(
                labelText: 'Filter by Category',
                border: OutlineInputBorder(),
              ),
              items: ['All', ..._categories].map((category) {
                return DropdownMenuItem(
                  value: category,
                  child: Text(category),
                );
              }).toList(),
              onChanged: (value) {
                setState(() {
```

```
            _filterCategory = value!;
          });
        },
      ),
      const SizedBox(height: 8),
      // Show/Hide Completed Toggle
      Row(
        children: [
          Checkbox(
            value: _showCompleted,
            onChanged: (value) {
              setState(() {
                _showCompleted = value!;
              });
            },
          ),
          const Text('Show completed tasks'),
        ],
      ),
    ],
  ),
),
// TODO List
Expanded(
  child: _filteredItems.isEmpty
      ? const Center(
          child: Text(
```

```
            'No TODO items found\nTap the + button to add your first task!',
            style: TextStyle(fontSize: 18, color: Colors.grey),
            textAlign: TextAlign.center,
          ),
        )
      : ListView.builder(
          itemCount: _filteredItems.length,
          itemBuilder: (context, index) {
            final item = _filteredItems[index];
            return Card(
              margin: const EdgeInsets.symmetric(
                horizontal: 16,
                vertical: 4,
              ),
              child: Padding(
                padding: const EdgeInsets.all(12.0),
                child: Column(
                  crossAxisAlignment: CrossAxisAlignment.start,
                  children: [
                    Row(
                      children: [
                        Checkbox(
                          value: item.isCompleted,
                          onChanged: (value) =>
                              _toggleTodoItem(item.id),
                        ),
                        Expanded(
```

```
                child: Text(
                  item.title,
                  style: TextStyle(
                    fontSize: 16,
                    fontWeight: FontWeight.w500,
                    decoration: item.isCompleted
                        ? TextDecoration.lineThrough
                        : null,
                  ),
                ),
              ),
              IconButton(
                icon: const Icon(
                  Icons.delete,
                  color: Colors.red,
                ),
                onPressed: () => _deleteTodoItem(item.id),
              ),
            ],
          ),
          if (item.description.isNotEmpty) ...[
            const SizedBox(height: 8),
            Text(
              item.description,
              style: const TextStyle(
                fontSize: 14,
                color: Colors.grey,
```

```
    ),
  ),
],
const SizedBox(height: 8),
Row(
  children: [
    Container(
      padding: const EdgeInsets.symmetric(
        horizontal: 8,
        vertical: 4,
      ),
      decoration: BoxDecoration(
        color: _getCategoryColor(item.category),
        borderRadius: BorderRadius.circular(12),
      ),
      child: Text(
        item.category,
        style: const TextStyle(
          color: Colors.white,
          fontSize: 12,
          fontWeight: FontWeight.w500,
        ),
      ),
    ),
    const SizedBox(width: 8),
    Text(
      'Created: ${_formatDate(item.createdAt)}',
```

```
                            style: const TextStyle(
                              fontSize: 12,
                              color: Colors.grey,
                            ),
                          ),
                        ],
                      ),
                    ],
                  ),
                ),
              );
            },
          ),
        ),
      ],
    ),
    floatingActionButton: FloatingActionButton(
      onPressed: () => _showAddTodoDialog(),
      tooltip: 'Add TODO',
      child: const Icon(Icons.add),
    ),
  );
}

Color _getCategoryColor(String category) {
  switch (category) {
    case 'Personal':
```

```
      return Colors.blue;
    case 'Work':
      return Colors.orange;
    case 'Shopping':
      return Colors.green;
    case 'Health':
      return Colors.red;
    case 'Education':
      return Colors.purple;
    default:
      return Colors.grey;
  }
}

String _formatDate(DateTime date) {
  return '${date.day}/${date.month}/${date.year}';
}

void _showAddTodoDialog() {
  showDialog(
    context: context,
    builder: (context) => AlertDialog(
      title: const Text('Add New TODO'),
      content: Column(
        mainAxisSize: MainAxisSize.min,
        children: [
          TextField(
```

```
  controller: _titleController,
  decoration: const InputDecoration(
    labelText: 'Title',
    border: OutlineInputBorder(),
  ),
),
const SizedBox(height: 16),
TextField(
  controller: _descriptionController,
  decoration: const InputDecoration(
    labelText: 'Description',
    border: OutlineInputBorder(),
  ),
  maxLines: 3,
),
const SizedBox(height: 16),
DropdownButtonFormField<String>(
  value: _selectedCategory,
  decoration: const InputDecoration(
    labelText: 'Category',
    border: OutlineInputBorder(),
  ),
  items: _categories.map((category) {
    return DropdownMenuItem(value: category, child: Text(category));
  }).toList(),
  onChanged: (value) {
    setState(() {
```

```
                _selectedCategory = value!;
              });
            },
          ),
        ],
      ),
      actions: [
        TextButton(
          onPressed: () => Navigator.pop(context),
          child: const Text('Cancel'),
        ),
        ElevatedButton(onPressed: _addTodoItem, child: const Text('Add')),
      ],
    ),
  );
 }

 @override
 void dispose() {
  _titleController.dispose();
  _descriptionController.dispose();
  super.dispose();
 }
}
```

OUTPUT:

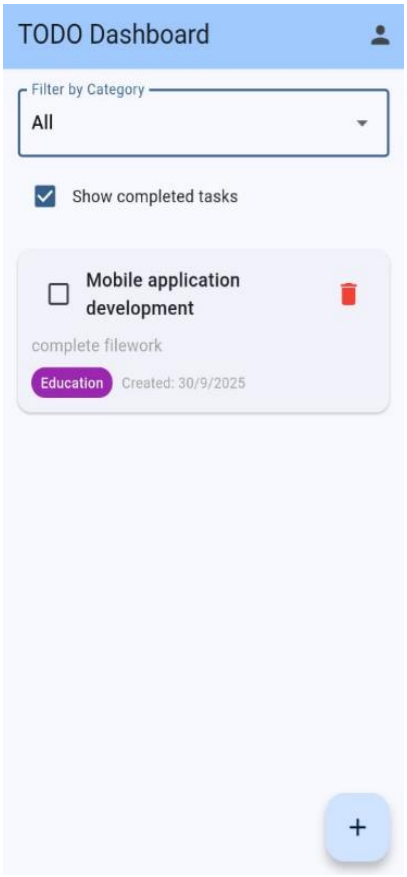


Fig. 1 todo list

LATEST APPLICATIONS:

latest applications of setstate and list view are endless, be it social media, or erp systems where thousands of data is displayed on screen, listview helps in organising

LEARNING OUTCOME:

The real life usage of listview, listbuilder, setstate one state, loader for the use of actual applications on internet.

REFERENCES:

- **Flutter Cookbook: Adding interactivity to your app** – https://docs.flutter.dev/cookbook/interactive
- **Medium – Flutter TODO App Tutorial using setState** – https://medium.com/flutter-community/flutter-todo-app-using-setstate-df3c1b1f5a8

- **YouTube – Flutter Tutorial: Build a TODO App with setState** – https://www.youtube.com/watch?v=1gDhl4leEzA

PRACTICAL:4

AIM:

.You are building a mobile application where users navigate through multiple screens like login, dashboard, and profile. Design a Form-based Registration App with validation using TextFormField.

THEORY:

String Manipulations as well as json form handling are very important topics in dart and flutter and help reduce server side costs during production of the application.

CODE:

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


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


  @override
  _RegistrationPageState createState() => _RegistrationPageState();
}


class _RegistrationPageState extends State<RegistrationPage> {
  final GlobalKey<FormState> _formKey = GlobalKey<FormState>();
  final TextEditingController _firstNameController = TextEditingController();
  final TextEditingController _lastNameController = TextEditingController();
  final TextEditingController _emailController = TextEditingController();
  final TextEditingController _phoneController = TextEditingController();
  final TextEditingController _passwordController = TextEditingController();
```

```
final TextEditingController _confirmPasswordController = TextEditingController();


bool _obscurePassword = true;
bool _obscureConfirmPassword = true;
bool _agreeToTerms = false;
String? _selectedGender;


@override
void dispose() {
  _firstNameController.dispose();
  _lastNameController.dispose();
  _emailController.dispose();
  _phoneController.dispose();
  _passwordController.dispose();
  _confirmPasswordController.dispose();
  super.dispose();
}


String? _validateName(String? value) {
  if (value == null || value.isEmpty) {
    return 'This field is required';
  }
  if (value.length < 2) {
    return 'Name must be at least 2 characters';
  }
  if (!RegExp(r'^[a-zA-Z\s]+$').hasMatch(value)) {
    return 'Name should only contain letters';
```

```
  }
  return null;
}

String? _validateEmail(String? value) {
  if (value == null || value.isEmpty) {
    return 'Please enter your email';
  }
  final emailRegExp = RegExp(r'^[\w-\.]+@([\w-]+\.)+[\w-]{2,4}$');
  if (!emailRegExp.hasMatch(value)) {
    return 'Please enter a valid email address';
  }
  return null;
}

String? _validatePhone(String? value) {
  if (value == null || value.isEmpty) {
    return 'Please enter your phone number';
  }
  if (!RegExp(r'^[0-9]{10}$').hasMatch(value)) {
    return 'Please enter a valid 10-digit phone number';
  }
  return null;
}

String? _validatePassword(String? value) {
  if (value == null || value.isEmpty) {
```

```
    return 'Please enter your password';
  }
  if (value.length < 8) {
    return 'Password must be at least 8 characters';
  }
  if (!RegExp(r'^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)').hasMatch(value)) {
    return 'Password must contain uppercase, lowercase, and number';
  }
  return null;
}

String? _validateConfirmPassword(String? value) {
  if (value == null || value.isEmpty) {
    return 'Please confirm your password';
  }
  if (value != _passwordController.text) {
    return 'Passwords do not match';
  }
  return null;
}

void _register() {
  if (_formKey.currentState!.validate()) {
    if (!_agreeToTerms) {
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(content: Text('Please agree to terms and conditions')),
      );
```

```
      return;
    }
    if (_selectedGender == null) {
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(content: Text('Please select your gender')),
      );
      return;
    }

    // Registration successful
    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(content: Text('Registration successful!')),
    );
    Navigator.pushReplacementNamed(context, '/dashboard');
  }
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text('Register'),
      backgroundColor: Colors.green,
    ),
    body: SingleChildScrollView(
      padding: EdgeInsets.all(16.0),
      child: Form(
```

```
key: _formKey,
child: Column(
  children: [
    Icon(
      Icons.person_add,
      size: 60,
      color: Colors.green,
    ),
    SizedBox(height: 20),

    // First Name
    TextFormField(
      controller: _firstNameController,
      decoration: InputDecoration(
        labelText: 'First Name',
        prefixIcon: Icon(Icons.person),
        border: OutlineInputBorder(borderRadius: BorderRadius.circular(10)),
      ),
      validator: _validateName,
    ),
    SizedBox(height: 16),

    // Last Name
    TextFormField(
      controller: _lastNameController,
      decoration: InputDecoration(
        labelText: 'Last Name',
```

```
      prefixIcon: Icon(Icons.person_outline),
      border: OutlineInputBorder(borderRadius: BorderRadius.circular(10)),
    ),
    validator: _validateName,
  ),
  SizedBox(height: 16),

  // Email
  TextFormField(
    controller: _emailController,
    keyboardType: TextInputType.emailAddress,
    decoration: InputDecoration(
      labelText: 'Email',
      prefixIcon: Icon(Icons.email),
      border: OutlineInputBorder(borderRadius: BorderRadius.circular(10)),
    ),
    validator: _validateEmail,
  ),
  SizedBox(height: 16),

  // Phone
  TextFormField(
    controller: _phoneController,
    keyboardType: TextInputType.phone,
    decoration: InputDecoration(
      labelText: 'Phone Number',
      prefixIcon: Icon(Icons.phone),
```

```
      border: OutlineInputBorder(borderRadius: BorderRadius.circular(10)),
    ),
    validator: _validatePhone,
  ),
  SizedBox(height: 16),

  // Gender Dropdown
  DropdownButtonFormField<String>(
    value: _selectedGender,
    decoration: InputDecoration(
      labelText: 'Gender',
      prefixIcon: Icon(Icons.people),
      border: OutlineInputBorder(borderRadius: BorderRadius.circular(10)),
    ),
    items: ['Male', 'Female', 'Other'].map((String value) {
      return DropdownMenuItem<String>(
        value: value,
        child: Text(value),
      );
    }).toList(),
    onChanged: (String? newValue) {
      setState(() {
        _selectedGender = newValue;
      });
    },
  ),
  SizedBox(height: 16),
```

```
// Password
TextFormField(
  controller: _passwordController,
  obscureText: _obscurePassword,
  decoration: InputDecoration(
    labelText: 'Password',
    prefixIcon: Icon(Icons.lock),
    suffixIcon: IconButton(
      icon: Icon(_obscurePassword ? Icons.visibility : Icons.visibility_off),
      onPressed: () {
        setState(() {
          _obscurePassword = !_obscurePassword;
        });
      },
    ),
    border: OutlineInputBorder(borderRadius: BorderRadius.circular(10)),
  ),
  validator: _validatePassword,
),
SizedBox(height: 16),

// Confirm Password
TextFormField(
  controller: _confirmPasswordController,
  obscureText: _obscureConfirmPassword,
  decoration: InputDecoration(
```

```
    labelText: 'Confirm Password',
    prefixIcon: Icon(Icons.lock_outline),
    suffixIcon: IconButton(
     icon: Icon(_obscureConfirmPassword ? Icons.visibility : Icons.visibility_off),
     onPressed: () {
      setState(() {
       _obscureConfirmPassword = !_obscureConfirmPassword;
      });
     },
    ),
    border: OutlineInputBorder(borderRadius: BorderRadius.circular(10)),
   ),
   validator: _validateConfirmPassword,
  ),
  SizedBox(height: 16),

  // Terms and Conditions
  Row(
   children: [
    Checkbox(
     value: _agreeToTerms,
     onChanged: (bool? value) {
      setState(() {
       _agreeToTerms = value ?? false;
      });
     },
    ),
```

```
      Expanded(
        child: Text(
          'I agree to the Terms and Conditions',
          style: TextStyle(fontSize: 14),
        ),
      ),
    ],
  ),
  SizedBox(height: 24),

  // Register Button
  ElevatedButton(
    onPressed: _register,
    style: ElevatedButton.styleFrom(
      backgroundColor: Colors.green,
      foregroundColor: Colors.white,
      shape: RoundedRectangleBorder(borderRadius: BorderRadius.circular(10)),
    ),
    child: Padding(
      padding: EdgeInsets.symmetric(horizontal: 32, vertical: 12),
      child: Text('REGISTER', style: TextStyle(fontSize: 16)),
    ),
  ),
  SizedBox(height: 16),

  TextButton(
    onPressed: () {
```

```
                Navigator.pop(context);
              },
              child: Text('Already have an account? Login here'),
            ),
          ],
        ),
      ),
    ),
  );
 }
}
```

OUTPUT:

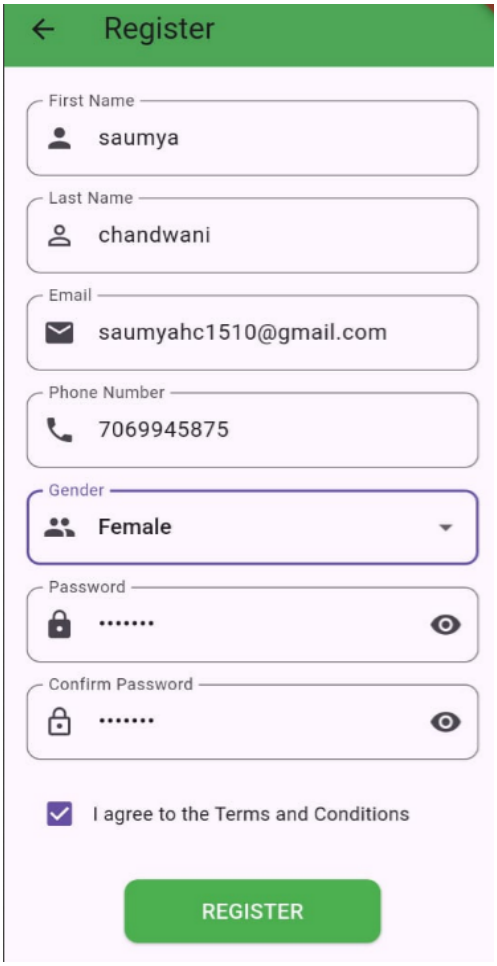


Fig 1. Form validation on registration page

LATEST APPLICATIONS:

the use of textfield and form validation is necessary in upcoming applications to reduce bot attacts and intrusion by automated software.

LEARNING OUTCOME:

Learnt the usage of text field and form validation for real life applications such as erp portals or banking systems.

REFERENCES:

- **Flutter Docs – Material Components (TextFormField & Button)** – https://docs.flutter.dev/development/ui/widgets/material
- **Medium – Flutter Form Validation Tutorial** – https://medium.com/flutter-community/flutter-form-validation-tutorial-9b7c36f32f1c
- **YouTube – Flutter Form Validation Tutorial** – https://www.youtube.com/watch?v=Z8hz0z9F3tI

PRACTICAL: 5

AIM:

You are building a mobile application where users navigate through multiple screens like login, dashboard, and profile. Build a Student Records App with CRUD operations using SQLite.

THEORY:

Allows Flutter apps to store, query, update, and delete structured data locally on the device.

Uses SQL (Structured Query Language) for database operations.

Supports tables, indexes, and relationships.

Data persists even when the app is closed or the device is restarted.

CODE:

```
import 'package:flutter/material.dart';
import '../database/database_helper.dart';
import '../models/student.dart';

class AddEditStudentScreen extends StatefulWidget {
  final Student? student;

  const AddEditStudentScreen({super.key, this.student});

  @override
  State<AddEditStudentScreen> createState() => _AddEditStudentScreenState();
}

class _AddEditStudentScreenState extends State<AddEditStudentScreen> {
  final _formKey = GlobalKey<FormState>();
  final DatabaseHelper _databaseHelper = DatabaseHelper();
```

```
late TextEditingController _nameController;
late TextEditingController _emailController;
late TextEditingController _phoneController;
late TextEditingController _gpaController;

String _selectedCourse = 'Computer Science';
DateTime _enrollmentDate = DateTime.now();
bool _isLoading = false;

final List<String> _courses = [
  'Computer Science',
  'Information Technology',
  'Data Science',
  'Software Engineering',
  'Cybersecurity',
  'Business Administration',
  'Marketing',
  'Finance',
  'Psychology',
  'Biology',
];

@override
void initState() {
  super.initState();
  _initializeControllers();
}
```

```
void _initializeControllers() {
  if (widget.student != null) {
    // Edit mode
    _nameController = TextEditingController(text: widget.student!.name);
    _emailController = TextEditingController(text: widget.student!.email);
    _phoneController = TextEditingController(text: widget.student!.phone);
    _gpaController = TextEditingController(text: widget.student!.gpa.toString());
    _selectedCourse = widget.student!.course;
    _enrollmentDate = widget.student!.enrollmentDate;
  } else {
    // Add mode
    _nameController = TextEditingController();
    _emailController = TextEditingController();
    _phoneController = TextEditingController();
    _gpaController = TextEditingController();
  }
}

@override
void dispose() {
  _nameController.dispose();
  _emailController.dispose();
  _phoneController.dispose();
  _gpaController.dispose();
  super.dispose();
}
```

```
Future<void> _selectDate() async {
  final DateTime? picked = await showDatePicker(
    context: context,
    initialDate: _enrollmentDate,
    firstDate: DateTime(2000),
    lastDate: DateTime.now(),
    builder: (context, child) {
      return Theme(
        data: Theme.of(context).copyWith(
          colorScheme: ColorScheme.light(
            primary: Colors.blue.shade600,
          ),
        ),
        child: child!,
      );
    },
  );

  if (picked != null && picked != _enrollmentDate) {
    setState(() {
      _enrollmentDate = picked;
    });
  }
}

Future<void> _saveStudent() async {
```

```
  if (!_formKey.currentState!.validate()) return;

  setState(() {
    _isLoading = true;
  });

  try {
    final student = Student(
      id: widget.student?.id,
      name: _nameController.text.trim(),
      email: _emailController.text.trim(),
      phone: _phoneController.text.trim(),
      course: _selectedCourse,
      gpa: double.parse(_gpaController.text.trim()),
      enrollmentDate: _enrollmentDate,
    );

    if (widget.student == null) {
      // Add new student
      await _databaseHelper.insertStudent(student);
      ScaffoldMessenger.of(context).showSnackBar(
        const SnackBar(
          content: Text('Student added successfully!'),
          backgroundColor: Colors.green,
        ),
      );
    } else {
```

```
      // Update existing student
      await _databaseHelper.updateStudent(student);
      ScaffoldMessenger.of(context).showSnackBar(
        const SnackBar(
          content: Text('Student updated successfully!'),
          backgroundColor: Colors.green,
        ),
      );
    }

    Navigator.of(context).pop(true);
  } catch (e) {
    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(
        content: Text('Error saving student: $e'),
        backgroundColor: Colors.red,
      ),
    );
  } finally {
    setState(() {
      _isLoading = false;
    });
  }
}

@override
Widget build(BuildContext context) {
```

```
final isEditing = widget.student != null;


return Scaffold(
  appBar: AppBar(
    title: Text(isEditing ? 'Edit Student' : 'Add Student'),
    backgroundColor: Colors.blue.shade600,
    foregroundColor: Colors.white,
    elevation: 0,
    actions: [
      TextButton(
        onPressed: _isLoading ? null : _saveStudent,
        child: Text(
          'SAVE',
          style: TextStyle(
            color: Colors.white,
            fontWeight: FontWeight.bold,
          ),
        ),
      ),
    ],
  ),
  body: Container(
    decoration: BoxDecoration(
      gradient: LinearGradient(
        begin: Alignment.topCenter,
        end: Alignment.bottomCenter,
        colors: [
```

```
          Colors.blue.shade600,
          Colors.blue.shade50,
        ],
        stops: const [0.0, 0.3],
      ),
    ),
    child: SafeArea(
      child: SingleChildScrollView(
        padding: const EdgeInsets.all(16.0),
        child: Card(
          elevation: 8,
          shape: RoundedRectangleBorder(
            borderRadius: BorderRadius.circular(16),
          ),
          child: Padding(
            padding: const EdgeInsets.all(24.0),
            child: Form(
              key: _formKey,
              child: Column(
                crossAxisAlignment: CrossAxisAlignment.start,
                children: [
                  // Header
                  Row(
                    children: [
                      Container(
                        width: 50,
                        height: 50,
```

```
          decoration: BoxDecoration(
            color: Colors.blue.shade100,
            shape: BoxShape.circle,
          ),
          child: Icon(
            isEditing ? Icons.edit : Icons.person_add,
            color: Colors.blue.shade600,
            size: 24,
          ),
        ),
        const SizedBox(width: 16),
        Expanded(
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
              Text(
                isEditing ? 'Edit Student Info' : 'Add New Student',
                style: Theme.of(context).textTheme.headlineSmall?.copyWith(
                  fontWeight: FontWeight.bold,
                  color: Colors.blue.shade800,
                ),
              ),
              Text(
                isEditing ? 'Update student details' : 'Fill in the student information',
                style: TextStyle(color: Colors.grey.shade600),
              ),
            ],
```

```
        ),
      ),
    ],
  ),
  const SizedBox(height: 32),

  // Name Field
  TextFormField(
    controller: _nameController,
    decoration: InputDecoration(
      labelText: 'Full Name *',
      prefixIcon: const Icon(Icons.person),
      border: OutlineInputBorder(
        borderRadius: BorderRadius.circular(12),
      ),
      filled: true,
      fillColor: Colors.grey.shade50,
    ),
    validator: (value) {
      if (value == null || value.trim().isEmpty) {
        return 'Please enter student name';
      }
      if (value.trim().length < 2) {
        return 'Name must be at least 2 characters';
      }
      return null;
    },
```

```
  textCapitalization: TextCapitalization.words,
),
const SizedBox(height: 16),

// Email Field
TextFormField(
  controller: _emailController,
  keyboardType: TextInputType.emailAddress,
  decoration: InputDecoration(
    labelText: 'Email Address *',
    prefixIcon: const Icon(Icons.email),
    border: OutlineInputBorder(
      borderRadius: BorderRadius.circular(12),
    ),
    filled: true,
    fillColor: Colors.grey.shade50,
  ),
  validator: (value) {
    if (value == null || value.trim().isEmpty) {
      return 'Please enter email address';
    }
    final emailRegex = RegExp(r'^[\w-\.]+@([\w-]+\.)+[\w-]{2,4}$');
    if (!emailRegex.hasMatch(value.trim())) {
      return 'Please enter a valid email address';
    }
    return null;
  },
```

```
),
const SizedBox(height: 16),

// Phone Field
TextFormField(
  controller: _phoneController,
  keyboardType: TextInputType.phone,
  decoration: InputDecoration(
    labelText: 'Phone Number *',
    prefixIcon: const Icon(Icons.phone),
    border: OutlineInputBorder(
      borderRadius: BorderRadius.circular(12),
    ),
    filled: true,
    fillColor: Colors.grey.shade50,
  ),
  validator: (value) {
    if (value == null || value.trim().isEmpty) {
      return 'Please enter phone number';
    }
    if (value.trim().length < 10) {
      return 'Please enter a valid phone number';
    }
    return null;
  },
),
const SizedBox(height: 16),
```

```
// Course Dropdown
DropdownButtonFormField<String>(
  value: _selectedCourse,
  decoration: InputDecoration(
    labelText: 'Course *',
    prefixIcon: const Icon(Icons.school),
    border: OutlineInputBorder(
      borderRadius: BorderRadius.circular(12),
    ),
    filled: true,
    fillColor: Colors.grey.shade50,
  ),
  items: _courses.map((String course) {
    return DropdownMenuItem<String>(
      value: course,
      child: Text(course),
    );
  }).toList(),
  onChanged: (String? newValue) {
    if (newValue != null) {
      setState(() {
        _selectedCourse = newValue;
      });
    }
  },
  validator: (value) {
```

```
      if (value == null || value.isEmpty) {
        return 'Please select a course';
      }
      return null;
    },
  ),
  const SizedBox(height: 16),

  // GPA Field
  TextFormField(
    controller: _gpaController,
    keyboardType: TextInputType.number,
    decoration: InputDecoration(
      labelText: 'GPA *',
      prefixIcon: const Icon(Icons.star),
      border: OutlineInputBorder(
        borderRadius: BorderRadius.circular(12),
      ),
      filled: true,
      fillColor: Colors.grey.shade50,
      helperText: 'Enter GPA (0.0 - 4.0)',
    ),
    validator: (value) {
      if (value == null || value.trim().isEmpty) {
        return 'Please enter GPA';
      }
      final gpa = double.tryParse(value.trim());
```

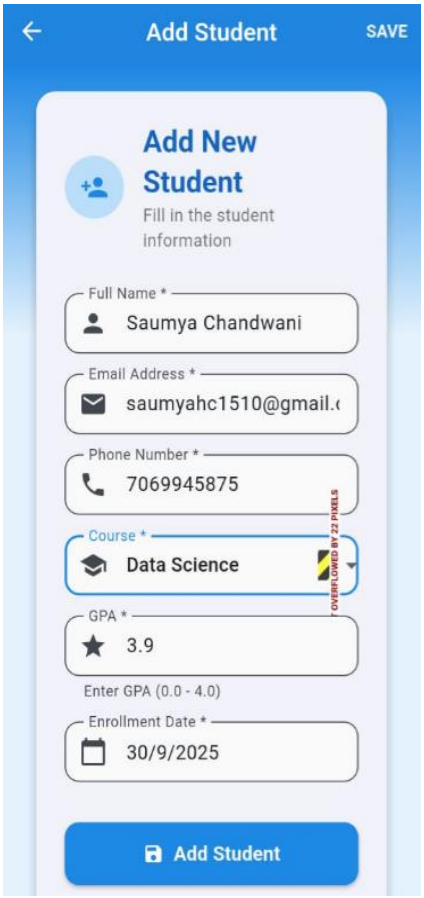
```
        if (gpa == null) {
          return 'Please enter a valid number';
        }
        if (gpa < 0.0 || gpa > 4.0) {
          return 'GPA must be between 0.0 and 4.0';
        }
        return null;
      },
    ),
    const SizedBox(height: 16),

    // Enrollment Date
    InkWell(
      onTap: _selectDate,
      child: InputDecorator(
        decoration: InputDecoration(
          labelText: 'Enrollment Date *',
          prefixIcon: const Icon(Icons.calendar_today),
          border: OutlineInputBorder(
            borderRadius: BorderRadius.circular(12),
          ),
          filled: true,
          fillColor: Colors.grey.shade50,
        ),
        child: Text(
          '${_enrollmentDate.day}/${_enrollmentDate.month}/${_enrollmentDate.year}',
          style: const TextStyle(fontSize: 16),
```

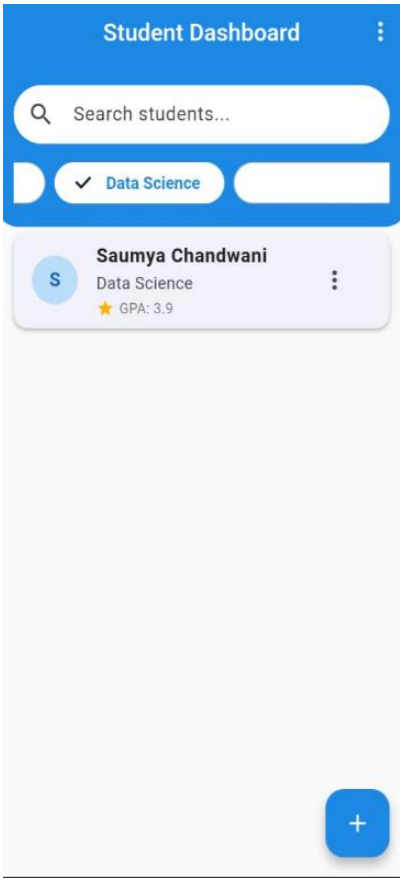
```
    ),
  ),
),
const SizedBox(height: 32),

// Save Button
SizedBox(
  width: double.infinity,
  height: 50,
  child: ElevatedButton(
    onPressed: _isLoading ? null : _saveStudent,
    style: ElevatedButton.styleFrom(
      backgroundColor: Colors.blue.shade600,
      foregroundColor: Colors.white,
      shape: RoundedRectangleBorder(
        borderRadius: BorderRadius.circular(12),
      ),
    ),
    child: _isLoading
        ? const CircularProgressIndicator(color: Colors.white)
        : Row(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
              Icon(isEditing ? Icons.update : Icons.save),
              const SizedBox(width: 8),
              Text(
                isEditing ? 'Update Student' : 'Add Student',
```

```
                    style: const TextStyle(
                      fontSize: 16,
                      fontWeight: FontWeight.bold,
                    ),
                  ),
                ],
              ),
            ),
          ),

          const SizedBox(height: 16),

          // Cancel Button
          SizedBox(
            width: double.infinity,
            height: 50,
            child: OutlinedButton(
              onPressed: _isLoading ? null : () => Navigator.of(context).pop(),
              style: OutlinedButton.styleFrom(
                side: BorderSide(color: Colors.grey.shade400),
                shape: RoundedRectangleBorder(
                  borderRadius: BorderRadius.circular(12),
                ),
              ),
              child: const Text(
                'Cancel',
                style: TextStyle(
```

```
                    fontSize: 16,
                    fontWeight: FontWeight.bold,
                  ),
                ),
              ),
            ),
          ],
        ),
      ),
    ),
  ),
 ),
),
),
);
}
}
```

OUTPUT:





LATEST APPLICATIONS:

Saving private keys in the device storage in encrypted form as part of persistent storage

LEARNING OUTCOME:

Learnt how to implement persistent storage in flutter which is important to save data which is going to be on the users device only

REFERENCES:

- **YouTube – Flutter SQLite CRUD Tutorial** – https://www.youtube.com/watch?v=Fq4PnsP3w2I
- **Medium – Build a Student Management App with Flutter & SQLite** – https://medium.com/flutter-community/build-student-management-app-with-flutter-sqlite-6bb5d8de4d5d

PRACTICAL: 6

AIM:

You are building a mobile application where users navigate through multiple screens like login, dashboard, and profile. Create a Notes App with persistent storage using Shared Preferences.

THEORY:

It allows us to store and retrieve small amounts of data (like strings, numbers, booleans, lists) persistently on the device.

Data is saved as key-value pairs and remains available even after the app is closed or the device is restarted.

It is best suited for storing user settings, preferences, login states, or other lightweight data.

CODE:

```
import 'package:flutter/material.dart';
import '../services/preferences_service.dart';
import '../models/note.dart';

/// Add Note Screen - Create new notes
/// Demonstrates creating and storing new data in SharedPreferences
class AddNoteScreen extends StatefulWidget {
  final PreferenceService preferenceService;
  final Note? existingNote;

  const AddNoteScreen({
    super.key,
    required this.preferenceService,
    this.existingNote,
  });

  @override
```

```
  State<AddNoteScreen> createState() => _AddNoteScreenState();
}

class _AddNoteScreenState extends State<AddNoteScreen> {
  final _formKey = GlobalKey<FormState>();
  final _titleController = TextEditingController();
  final _contentController = TextEditingController();
  final _tagController = TextEditingController();

  List<String> _tags = [];
  bool _isFavorite = false;
  bool _isLoading = false;

  @override
  void initState() {
    super.initState();
    if (widget.existingNote != null) {
      _titleController.text = widget.existingNote!.title;
      _contentController.text = widget.existingNote!.content;
      _tags = List.from(widget.existingNote!.tags);
      _isFavorite = widget.existingNote!.isFavorite;
    }
  }

  void _addTag() {
    final tag = _tagController.text.trim().toLowerCase();
    if (tag.isNotEmpty && !_tags.contains(tag)) {
```

```
    setState(() {
      _tags.add(tag);
      _tagController.clear();
    });
  }
}

void _removeTag(String tag) {
  setState(() => _tags.remove(tag));
}

Future<void> _saveNote() async {
  if (!_formKey.currentState!.validate()) return;

  setState(() => _isLoading = true);

  final now = DateTime.now();
  final note = Note(
    id: widget.existingNote?.id ?? now.millisecondsSinceEpoch.toString(),
    title: _titleController.text.trim(),
    content: _contentController.text.trim(),
    createdAt: widget.existingNote?.createdAt ?? now,
    updatedAt: now,
    isFavorite: _isFavorite,
    tags: _tags,
  );
```

```
  // Simulate saving process
  await Future.delayed(const Duration(milliseconds: 500));


  Navigator.pop(context, note);
 }


 void _showDiscardDialog() {
  final hasChanges = _titleController.text.trim().isNotEmpty ||
                _contentController.text.trim().isNotEmpty ||
                _tags.isNotEmpty;


  if (!hasChanges) {
   Navigator.pop(context);
   return;
  }


  showDialog(
   context: context,
   builder: (context) => AlertDialog(
    title: const Text('Discard Changes?'),
    content: const Text('You have unsaved changes. Are you sure you want to discard them?'),
    actions: [
     TextButton(
      onPressed: () => Navigator.pop(context),
      child: const Text('Cancel'),
     ),
     TextButton(
```

```
          onPressed: () {
            Navigator.pop(context); // Close dialog
            Navigator.pop(context); // Close screen
          },
          child: const Text('Discard'),
        ),
      ],
    ),
  );
}

@override
Widget build(BuildContext context) {
  final isEditing = widget.existingNote != null;

  return WillPopScope(
    onWillPop: () async {
      _showDiscardDialog();
      return false;
    },
    child: Scaffold(
      appBar: AppBar(
        title: Text(isEditing ? 'Edit Note' : 'New Note'),
        leading: IconButton(
          icon: const Icon(Icons.close),
          onPressed: _showDiscardDialog,
        ),
```

```
      actions: [
        IconButton(
          icon: Icon(_isFavorite ? Icons.favorite : Icons.favorite_border),
          onPressed: () => setState(() => _isFavorite = !_isFavorite),
        ),
        TextButton(
          onPressed: _isLoading ? null : _saveNote,
          child: _isLoading
            ? const SizedBox(
                width: 16,
                height: 16,
                child: CircularProgressIndicator(strokeWidth: 2),
              )
            : const Text('Save'),
        ),
      ],
    ),
    body: Form(
      key: _formKey,
      child: SingleChildScrollView(
        padding: const EdgeInsets.all(16.0),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            // Title Field
            TextFormField(
              controller: _titleController,
```

```
    decoration: const InputDecoration(
      labelText: 'Title',
      hintText: 'Enter note title...',
    ),
    style: const TextStyle(
      fontSize: 18,
      fontWeight: FontWeight.bold,
    ),
    validator: (value) {
      if (value == null || value.trim().isEmpty) {
        return 'Please enter a title';
      }
      return null;
    },
    textInputAction: TextInputAction.next,
  ),
  const SizedBox(height: 16),

  // Content Field
  TextFormField(
    controller: _contentController,
    decoration: const InputDecoration(
      labelText: 'Content',
      hintText: 'Write your note here...',
      border: OutlineInputBorder(),
    ),
    maxLines: 10,
```

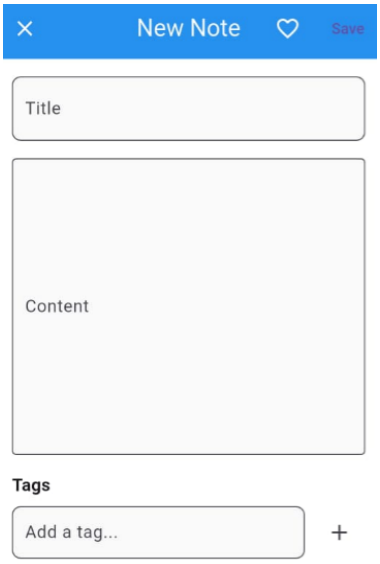
```
  validator: (value) {
    if (value == null || value.trim().isEmpty) {
      return 'Please enter some content';
    }
    return null;
  },
  textInputAction: TextInputAction.newline,
),
const SizedBox(height: 16),

// Tags Section
const Text(
  'Tags',
  style: TextStyle(
    fontSize: 16,
    fontWeight: FontWeight.bold,
  ),
),
const SizedBox(height: 8),

// Add Tag Field
Row(
  children: [
    Expanded(
      child: TextFormField(
        controller: _tagController,
        decoration: const InputDecoration(
```

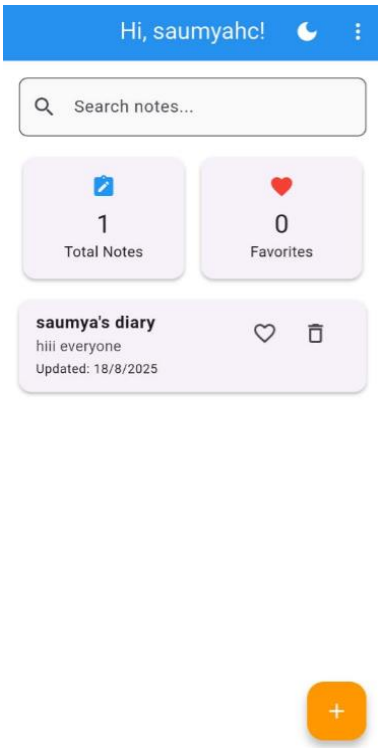
```
            hintText: 'Add a tag...',
            isDense: true,
          ),
          onFieldSubmitted: (_) => _addTag(),
          textInputAction: TextInputAction.done,
        ),
      ),
      const SizedBox(width: 8),
      IconButton(
        onPressed: _addTag,
        icon: const Icon(Icons.add),
      ),
    ],
  ),
  const SizedBox(height: 8),

  // Tags Display
  if (_tags.isNotEmpty)
    Wrap(
      spacing: 8,
      runSpacing: 8,
      children: _tags.map((tag) =>
        Chip(
          label: Text(tag),
          onDeleted: () => _removeTag(tag),
          deleteIcon: const Icon(Icons.close, size: 18),
        ),
```

```
            ).toList(),
          ),
        const SizedBox(height: 24),

        // Note Info (if editing)
        if (isEditing)
          Card(
            child: Padding(
              padding: const EdgeInsets.all(16.0),
              child: Column(
                crossAxisAlignment: CrossAxisAlignment.start,
                children: [
                  const Text(
                    'Note Information',
                    style: TextStyle(fontWeight: FontWeight.bold),
                  ),
                  const SizedBox(height: 8),
                  Text('Created: ${_formatDateTime(widget.existingNote!.createdAt)}'),
                  Text('Last Updated: ${_formatDateTime(widget.existingNote!.updatedAt)}'),
                  Text('ID: ${widget.existingNote!.id}'),
                ],
              ),
            ),
          ),
      ],
    ),
  ),
```

```
        ),
      ),
    );
  }

  String _formatDateTime(DateTime dateTime) {
    return '${dateTime.day}/${dateTime.month}/${dateTime.year} at
${dateTime.hour}:${dateTime.minute.toString().padLeft(2, '0')}';
  }

  @override
  void dispose() {
    _titleController.dispose();
    _contentController.dispose();
    _tagController.dispose();
    super.dispose();
  }
}
```

OUTPUT:





LATEST APPLICATIONS:

Shared preferences are constantly being used for storing user api tokens and login credentials of users.

LEARNING OUTCOMES:

You will understand how to implement persistent local storage in Flutter using Shared Preferences, enabling apps to save and retrieve user data like notes across sessions while managing multi-screen navigation.

REFERENCES:

- **Flutter Cookbook: Store simple data using Shared Preferences** – https://docs.flutter.dev/cookbook/persistence/key-value
- **Medium – Flutter Notes App using Shared Preferences** – https://medium.com/flutter-community/flutter-notes-app-using-shared-preferences-2c80a6a3fba1
- **YouTube – Flutter Notes App with Shared Preferences** – https://www.youtube.com/watch?v=G8mO1Bds06A

PRACTICAL: 7

AIM:

You are building a mobile application where users navigate through multiple screens like login, dashboard, and profile. Design a Product Catalog App using GridView and custom cards with images.

THEORY:

Grid view is used to structure list content on the ui in a more beautiful manner, helps us make applications such as gallery, shopping product catalogs, or social media applications

CODE:

```
// Product screen
// screens/catalog_screen.dart
import 'package:flutter/material.dart';
import '../models/product.dart';
import '../services/data_service.dart';
import '../widgets/custom_grid_view.dart';
import 'product_detail_screen.dart';

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

  @override
  State<CatalogScreen> createState() => _CatalogScreenState();
}

class _CatalogScreenState extends State<CatalogScreen>
    with TickerProviderStateMixin {
```

```
List<Product> _products = [];
List<Product> _filteredProducts = [];
String _selectedCategory = 'All';
bool _isGridView = true;
final TextEditingController _searchController = TextEditingController();
late AnimationController _animationController;
late Animation<double> _animation;

@override
void initState() {
  super.initState();
  _loadProducts();

  _animationController = AnimationController(
    duration: const Duration(milliseconds: 300),
    vsync: this,
  );
  _animation = CurvedAnimation(
    parent: _animationController,
    curve: Curves.easeInOut,
  );
  _animationController.forward();
}

@override
void dispose() {
  _searchController.dispose();
```

```
  _animationController.dispose();
  super.dispose();
 }

 void _loadProducts() {
  setState(() {
    _products = DataService.getAllProducts();
    _filteredProducts = _products;
  });
 }

 void _filterProducts() {
  setState(() {
    List<Product> filtered = _products;

    // Filter by category
    if (_selectedCategory != 'All') {
      filtered = filtered
          .where((product) => product.category == _selectedCategory)
          .toList();
    }

    // Filter by search query
    final query = _searchController.text.trim();
    if (query.isNotEmpty) {
      filtered = DataService.searchProducts(query)
          .where((product) =>
```

```
            _selectedCategory == 'All' ||
            product.category == _selectedCategory)
        .toList();
    }

    _filteredProducts = filtered;
  });
}

void _onProductTap(Product product) {
  Navigator.push(
    context,
    PageRouteBuilder(
      pageBuilder: (context, animation, secondaryAnimation) =>
          ProductDetailScreen(product: product),
      transitionsBuilder: (context, animation, secondaryAnimation, child) {
        return SlideTransition(
          position: Tween<Offset>(
            begin: const Offset(1.0, 0.0),
            end: Offset.zero,
          ).animate(animation),
          child: child,
        );
      },
    ),
  );
}
```

```
@override
Widget build(BuildContext context) {
  final categories = ['All', ...DataService.getCategories()];


  return Scaffold(
    backgroundColor: Colors.grey[50],
    appBar: AppBar(
      title: const Text('Product Catalog'),
      backgroundColor: Colors.white,
      foregroundColor: Colors.black,
      elevation: 0,
      actions: [
        IconButton(
          icon: Icon(_isGridView ? Icons.view_list : Icons.grid_view),
          onPressed: () {
            setState(() {
              _isGridView = !_isGridView;
            });
          },
        ),
        const SizedBox(width: 8),
      ],
      bottom: PreferredSize(
        preferredSize: const Size.fromHeight(1),
        child: Container(
          height: 1,
```

```
        color: Colors.grey[300],
      ),
    ),
  ),
  body: FadeTransition(
    opacity: _animation,
    child: Column(
      children: [
        // Search and Filter Section
        Container(
          padding: const EdgeInsets.all(16),
          color: Colors.white,
          child: Column(
            children: [
              // Search Bar
              TextField(
                controller: _searchController,
                onChanged: (_) => _filterProducts(),
                decoration: InputDecoration(
                  hintText: 'Search products...',
                  prefixIcon: const Icon(Icons.search),
                  border: OutlineInputBorder(
                    borderRadius: BorderRadius.circular(12),
                    borderSide: BorderSide(color: Colors.grey[300]!),
                  ),
                  enabledBorder: OutlineInputBorder(
                    borderRadius: BorderRadius.circular(12),
```

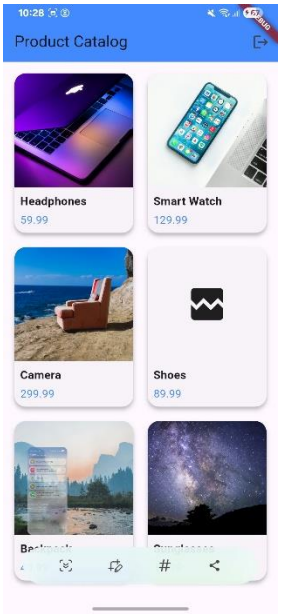
```
          borderSide: BorderSide(color: Colors.grey[300]!),
        ),
        filled: true,
        fillColor: Colors.grey[50],
      ),
    ),
    const SizedBox(height: 16),
    // Category Filter
    SizedBox(
      height: 40,
      child: ListView.builder(
        scrollDirection: Axis.horizontal,
        itemCount: categories.length,
        itemBuilder: (context, index) {
          final category = categories[index];
          final isSelected = category == _selectedCategory;

          return Padding(
            padding: const EdgeInsets.only(right: 8),
            child: FilterChip(
              label: Text(category),
              selected: isSelected,
              onSelected: (selected) {
                setState(() {
                  _selectedCategory = category;
                  _filterProducts();
                });
```

```
                },
                backgroundColor: Colors.grey[100],
                selectedColor: Theme.of(context).primaryColor,
                labelStyle: TextStyle(
                  color: isSelected ? Colors.white : Colors.black,
                  fontWeight: isSelected
                      ? FontWeight.bold
                      : FontWeight.normal,
                ),
              ),
            );
          },
        ),
      ),
    ],
  ),
),
// Results Count
Container(
  padding: const EdgeInsets.symmetric(horizontal: 16, vertical: 8),
  child: Row(
    children: [
      Text(
        '${_filteredProducts.length} products found',
        style: TextStyle(
          color: Colors.grey[600],
          fontSize: 14,
```

```
                ),
              ),
            ],
          ),
        ),
        // Products Grid/List
        Expanded(
          child: _isGridView
              ? ResponsiveGridView(
                  products: _filteredProducts,
                  onProductTap: _onProductTap,
                )
              : CustomGridView(
                  products: _filteredProducts,
                  onProductTap: _onProductTap,
                  crossAxisCount: 1,
                  childAspectRatio: 3,
                  mainAxisSpacing: 8,
                  crossAxisSpacing: 0,
                  padding: const EdgeInsets.all(16),
                ),
        ),
      ],
    ),
  ),
 );
}
```

```
}
```

OUTPUT:



LATEST APPLICATIONS:

Applications include making shopping applications, payment applications.

LEARNING OUTCOME:

You will learn how to display dynamic content in a structured grid layout using GridView, create custom cards with images and text, and manage multi-screen navigation to build visually appealing product catalog apps in Flutter.

REFERENCES:

- **Medium – Flutter Product Catalog with GridView** – https://medium.com/flutter-community/flutter-product-catalog-ui-with-gridview-28a39b1b40c3
- **YouTube – Flutter GridView Product Catalog Tutorial** – https://www.youtube.com/watch?v=R19_gJULflY

PRACTICAL: 8

AIM:

You are building a mobile application where users navigate through multiple screens like login, dashboard, and profile. Connect to a REST API (e.g., weather, user data) and display using FutureBuilder.

THEORY:

FutureBuilder is a Flutter widget that helps you build UI based on the state of a Future (an asynchronous operation).

CODE:

```
//weather display screen

// screens/dashboard_screen.dart

import 'package:flutter/material.dart';

import '../services/api_service.dart';

import '../services/auth_service.dart';

import '../models/weather.dart';


class DashboardScreen extends StatefulWidget {

  @override

  _DashboardScreenState createState() => _DashboardScreenState();

}


class _DashboardScreenState extends State<DashboardScreen> {

  final _cityController = TextEditingController(text: 'London');

  late Future<Weather> _weatherFuture;


  @override

  void initState() {

    super.initState();
```

```
  _weatherFuture = ApiService.getWeather(_cityController.text);
}

void _searchWeather() {
  if (_cityController.text.trim().isNotEmpty) {
    setState(() {
      _weatherFuture = ApiService.getWeather(_cityController.text.trim());
    });
  }
}

Future<void> _logout() async {
  await AuthService.logout();
  Navigator.pushReplacementNamed(context, '/login');
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text('Weather Dashboard'),
      actions: [
        IconButton(
          icon: Icon(Icons.person),
          onPressed: () => Navigator.pushNamed(context, '/profile'),
        ),
        IconButton(icon: Icon(Icons.logout), onPressed: _logout),
```

```
    ],
  ),
  body: SingleChildScrollView(
    padding: EdgeInsets.all(16.0),
    child: Column(
      crossAxisAlignment: CrossAxisAlignment.start,
      children: [
        Text(
          'Weather Information',
          style: Theme.of(context).textTheme.headlineSmall,
        ),
        SizedBox(height: 16),
        Row(
          children: [
            Expanded(
              child: TextField(
                controller: _cityController,
                decoration: InputDecoration(
                  labelText: 'Enter city name',
                  border: OutlineInputBorder(),
                  prefixIcon: Icon(Icons.location_city),
                ),
                onSubmitted: (_) => _searchWeather(),
              ),
            ),
            SizedBox(width: 8),
            ElevatedButton(
```

```
            onPressed: _searchWeather,
            child: Text('Search'),
          ),
        ],
      ),
      SizedBox(height: 16),
      // FutureBuilder for handling async API calls with loading/error states
      FutureBuilder<Weather>(
        future: _weatherFuture,
        builder: (context, snapshot) {
          // Loading state
          if (snapshot.connectionState == ConnectionState.waiting) {
            return Card(
              elevation: 4,
              child: Padding(
                padding: EdgeInsets.all(32.0),
                child: Center(
                  child: Column(
                    children: [
                      CircularProgressIndicator(),
                      SizedBox(height: 16),
                      Text(
                        'Loading weather data...',
                        style: TextStyle(fontSize: 16),
                      ),
                    ],
                  ),
```

```
        ),
      ),
    );
  }

  // Error state
  if (snapshot.hasError) {
    return Card(
      elevation: 4,
      color: Colors.red[50],
      child: Padding(
        padding: EdgeInsets.all(16.0),
        child: Column(
          children: [
            Icon(
              Icons.error_outline,
              color: Colors.red,
              size: 48,
            ),
            SizedBox(height: 16),
            Text(
              'Oops! Something went wrong',
              style: TextStyle(
                fontSize: 18,
                fontWeight: FontWeight.bold,
                color: Colors.red[700],
              ),
```

```
            ),
            SizedBox(height: 8),
            Text(
              '${snapshot.error}',
              style: TextStyle(color: Colors.red[600]),
              textAlign: TextAlign.center,
            ),
            SizedBox(height: 16),
            ElevatedButton.icon(
              onPressed: _searchWeather,
              icon: Icon(Icons.refresh),
              label: Text('Try Again'),
              style: ElevatedButton.styleFrom(
                backgroundColor: Colors.red,
                foregroundColor: Colors.white,
              ),
            ),
          ],
        ),
      ),
    );
  }

  // Success state with data
  if (snapshot.hasData) {
    final weather = snapshot.data!;
    return Card(
```

```
elevation: 6,
child: Container(
 width: double.infinity,
 decoration: BoxDecoration(
  borderRadius: BorderRadius.circular(8),
  gradient: LinearGradient(
   begin: Alignment.topLeft,
   end: Alignment.bottomRight,
   colors: [Colors.blue[400]!, Colors.blue[600]!],
  ),
 ),
 child: Padding(
  padding: EdgeInsets.all(24.0),
  child: Column(
   crossAxisAlignment: CrossAxisAlignment.start,
   children: [
    Row(
     mainAxisAlignment: MainAxisAlignment.spaceBetween,
     children: [
      Column(
       crossAxisAlignment: CrossAxisAlignment.start,
       children: [
        Text(
         weather.cityName,
         style: TextStyle(
          fontSize: 28,
          fontWeight: FontWeight.bold,
```

```
            color: Colors.white,
          ),
        ),
        Text(
          weather.description.toUpperCase(),
          style: TextStyle(
            fontSize: 16,
            color: Colors.white70,
            fontWeight: FontWeight.w500,
          ),
        ),
      ],
    ),
    Column(
      crossAxisAlignment: CrossAxisAlignment.end,
      children: [
        Text(
          '${weather.temperature.toStringAsFixed(1)}°C',
          style: TextStyle(
            fontSize: 36,
            fontWeight: FontWeight.bold,
            color: Colors.white,
          ),
        ),
        Text(
          'Feels like ${weather.temperature.toStringAsFixed(0)}°C',
          style: TextStyle(
```

```
                fontSize: 14,
                color: Colors.white70,
              ),
            ),
          ],
        ),
      ],
    ),
    SizedBox(height: 24),
    Container(
      padding: EdgeInsets.all(16),
      decoration: BoxDecoration(
        color: Colors.white.withOpacity(0.2),
        borderRadius: BorderRadius.circular(8),
      ),
      child: Row(
        mainAxisAlignment:
            MainAxisAlignment.spaceAround,
        children: [
          _buildWeatherDetail(
            Icons.water_drop,
            '${weather.humidity.toInt()}%',
            'Humidity',
          ),
          Container(
            height: 40,
            width: 1,
```

```
                    color: Colors.white30,
                  ),
                  _buildWeatherDetail(
                    Icons.air,
                    '${weather.windSpeed.toStringAsFixed(1)} m/s',
                    'Wind Speed',
                  ),
                ],
              ),
            ),
          ],
        ),
      ),
    );
  }

  return SizedBox.shrink();
 },
),
SizedBox(height: 24),
Card(
 child: Padding(
  padding: EdgeInsets.all(16.0),
  child: Column(
   crossAxisAlignment: CrossAxisAlignment.start,
   children: [
```

```
                Text(
                  'API Information',
                  style: TextStyle(
                    fontSize: 18,
                    fontWeight: FontWeight.bold,
                  ),
                ),
                SizedBox(height: 8),
                Text(
                  'Weather data provided by OpenWeatherMap API',
                  style: TextStyle(color: Colors.grey[600]),
                ),
                Text(
                  'Enter any city name to get current weather information',
                  style: TextStyle(color: Colors.grey[600]),
                ),
              ],
            ),
          ),
        ),
      ],
    ),
  ),
 );
}

Widget _buildWeatherDetail(IconData icon, String value, String label) {
```

```
    return Column(
      children: [
        Icon(icon, color: Colors.white, size: 24),
        SizedBox(height: 4),
        Text(
          value,
          style: TextStyle(
            fontSize: 16,
            fontWeight: FontWeight.bold,
            color: Colors.white,
          ),
        ),
        Text(label, style: TextStyle(fontSize: 12, color: Colors.white70)),
      ],
    );
  }

  @override
  void dispose() {
    _cityController.dispose();
    super.dispose();
  }
}
```

OUTPUT:

LATEST APPLICATIONS:

I use FutureBuilder in my Flutter apps to display data from REST APIs, databases, or any asynchronous source in real time.

LEARNING OUTCOME:

I learnt how to use FutureBuilder to handle asynchronous data and update my app's UI dynamically in Flutter.

REFERENCES:

- **Flutter Docs – FutureBuilder Widget** – https://docs.flutter.dev/development/ui/widgets/async
- **Medium – Flutter REST API Tutorial using FutureBuilder** – https://medium.com/flutter-community/flutter-rest-api-tutorial-using-futurebuilder-124c13f6f889

PRACTICAL: 9

AIM:

You are building a mobile application where users navigate through multiple screens like login, dashboard, and profile. Develop a Login Authentication App using API-based credential check and session handling.

THEORY:

User Input:

The user enters their email address in the app (email_verification.dart).

OTP Request:

The app sends a POST request to the backend (send_otp.php) with the email.

The backend generates a One-Time Password (OTP) and sends it to the user's email.

OTP Verification:

The user enters the OTP in the app.

The app sends the OTP and email to the backend (verify_otp.php).

The backend checks if the OTP is correct and not expired.

Session Creation:

If the OTP is valid, the backend marks the user as authenticated and (optionally) creates a session token (e.g., JWT or random session ID).

CODE:

```
// auth_input_page.dart (Mobile App)

import 'dart:convert';

import 'package:flutter/material.dart';

import 'package:http/http.dart' as http;

import 'signup_page.dart';
```

```
import '../constants.dart';
import '../theme.dart';
import 'otp_verification_page.dart';

class EmailVerificationPage extends StatefulWidget {
  final String? initialEmail;
  const EmailVerificationPage({super.key, this.initialEmail});

  @override
  State<EmailVerificationPage> createState() => _EmailVerificationPageState();
}

class _EmailVerificationPageState extends State<EmailVerificationPage> {
  final TextEditingController _emailController = TextEditingController();
  String? _emailError;
  bool _isLoading = false;

  bool _isValidEmail(String email) {
    return RegExp(r'^[\w-\.]+@([\w-]+\.)+[\w-]{2,4}$').hasMatch(email);
  }

  @override
  void initState() {
    super.initState();
    if (widget.initialEmail != null) {
      _emailController.text = widget.initialEmail!;
    }
```

```
}

Future<void> _handleSendOtp() async {
  final email = _emailController.text.trim();
  if (!_isValidEmail(email)) {
    setState(() {
      _emailError = 'Please enter a valid email address';
    });
    return;
  }
  setState(() {
    _emailError = null;
    _isLoading = true;
  });
  try {
    final response = await http.post(
      Uri.parse(ApiConstants.sendOtp),
      headers: {'Content-Type': 'application/json'},
      body: jsonEncode({'email': email}),
    );
    final data = jsonDecode(response.body);
    if (data['success']) {
      ScaffoldMessenger.of(
        context,
      ).showSnackBar(SnackBar(content: Text('OTP sent! Check your email.')));
      Navigator.push(
        context,
```

```
        MaterialPageRoute(
          builder: (context) => OTPVerificationPage(email: email),
        ),
      );
    } else {
      setState(() {
        _emailError = data['message'] ?? 'Failed to send OTP';
      });
    }
  } catch (e) {
    setState(() {
      _emailError = 'Failed to send OTP: $e';
    });
  } finally {
    setState(() {
      _isLoading = false;
    });
  }
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    backgroundColor: Theme.of(context).scaffoldBackgroundColor,
    body: Center(
      child: SingleChildScrollView(
        child: Padding(
```

```
padding: const EdgeInsets.symmetric(horizontal: 24.0, vertical: 32),
child: Card(
 elevation: 8,
 shape: RoundedRectangleBorder(
  borderRadius: BorderRadius.circular(24),
 ),
 child: Padding(
  padding: const EdgeInsets.all(24.0),
  child: Column(
   mainAxisSize: MainAxisSize.min,
   children: [
    Icon(
     Icons.email_outlined,
     size: 64,
     color: AppColors.primary,
    ),
    const SizedBox(height: 24),
    Text(
     'Welcome Back',
     style: Theme.of(context).textTheme.displayLarge?.copyWith(
      fontSize: 26,
      color: AppColors.primary,
      fontWeight: FontWeight.bold,
     ),
    ),
    const SizedBox(height: 8),
    Text(
```

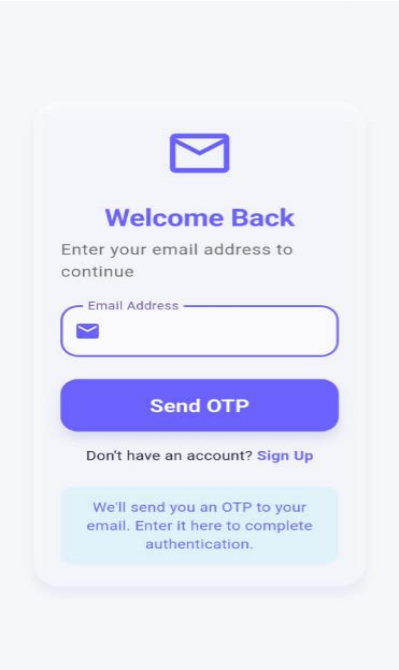
```
  'Enter your email address to continue',
  style: Theme.of(
    context,
  ).textTheme.bodyLarge?.copyWith(color: Colors.black54),
),
const SizedBox(height: 24),
TextField(
  controller: _emailController,
  keyboardType: TextInputType.emailAddress,
  decoration: InputDecoration(
    prefixIcon: Icon(Icons.email, color: AppColors.primary),
    labelText: 'Email Address',
    errorText: _emailError,
  ),
  onChanged: (val) {
    setState(() {
      if (_isValidEmail(val.trim())) {
        _emailError = null;
      }
    });
  },
),
const SizedBox(height: 24),
SizedBox(
  width: double.infinity,
  child: ElevatedButton(
    onPressed: _isLoading ? null : _handleSendOtp,
```

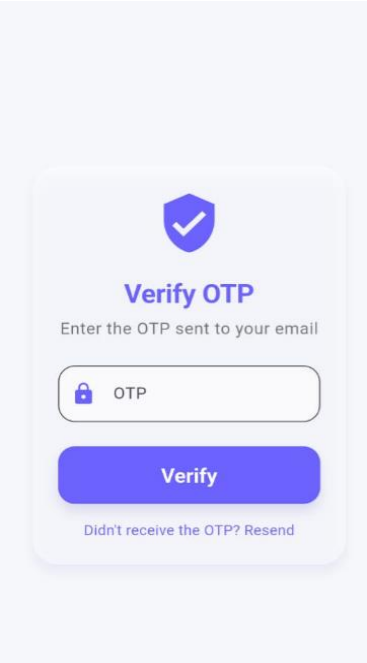
```
        style: ElevatedButton.styleFrom(
          padding: const EdgeInsets.symmetric(vertical: 16),
          backgroundColor: AppColors.primary,
          shape: RoundedRectangleBorder(
            borderRadius: BorderRadius.circular(16),
          ),
        ),
        child: _isLoading
            ? SizedBox(
                width: 24,
                height: 24,
                child: CircularProgressIndicator(
                  strokeWidth: 2,
                  color: Colors.white,
                ),
              )
            : Text(
                'Send OTP',
                style: Theme.of(context).textTheme.titleLarge
                    ?.copyWith(color: Colors.white),
              ),
      ),
    ),
    const SizedBox(height: 16),
    Row(
      mainAxisAlignment: MainAxisAlignment.center,
      children: [
```

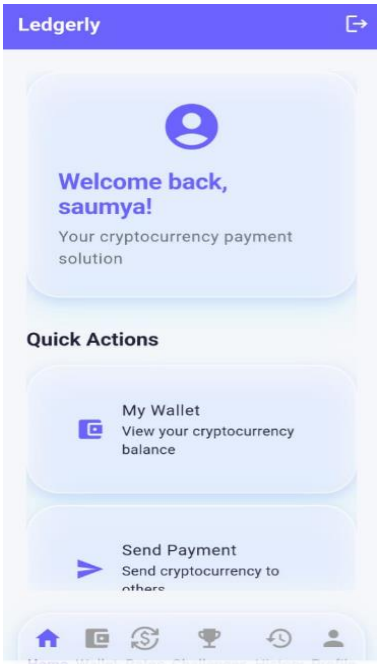
```
          Text(
            "Don't have an account? ",
            style: Theme.of(context).textTheme.bodyMedium,
          ),
          GestureDetector(
            onTap: () {
              Navigator.push(
                context,
                MaterialPageRoute(
                  builder: (context) => SignUpPage(),
                ),
              );
            },
            child: Text(
              'Sign Up',
              style: Theme.of(context).textTheme.bodyMedium
                  ?.copyWith(
                    color: AppColors.primary,
                    fontWeight: FontWeight.bold,
                  ),
            ),
          ),
        ],
      ),
      const SizedBox(height: 24),
      Container(
        width: double.infinity,
```

```
                padding: const EdgeInsets.all(12),
                decoration: BoxDecoration(
                  color: AppColors.secondary.withOpacity(0.08),
                  borderRadius: BorderRadius.circular(12),
                ),
                child: Text(
                  "We'll send you an OTP to your email. Enter it here to complete authentication.",
                  style: Theme.of(context).textTheme.bodyMedium?.copyWith(
                    color: AppColors.primary,
                  ),
                  textAlign: TextAlign.center,
                ),
              ),
            ],
          ),
        ),
      ),
    ),
  );
 }
}
```

OUTPUT:







LATEST APPLICATIONS:

Used in modern apps like **e-commerce, banking, and social media platforms** for secure login and session management.

LEARNING OUTCOME:

Understand API-based authentication, secure session handling, and Flutter state management for login flows.

REFERENCES:

1. **Medium – Flutter API-based Login Authentication** – https://medium.com/flutter-community/flutter-login-authentication-using-rest-api-8c7592c9e3a2
2. **YouTube – Flutter REST API Login & Session Handling** – https://www.youtube.com/watch?v=f2cY4yix5Wk

PRACTICAL: 10

AIM:

You are building a mobile application where users navigate through multiple screens like login, dashboard, and profile. Create and generate a Signed APK for deployment. Document steps to publish the app.

THEORY:

Generate a Keystore

Open a terminal and run:

Follow the prompts and remember your password and alias.

Move the Keystore to Your Project

Copy my-release-key.jks to your Flutter project's android/app directory.

Configure the Keystore in Gradle

CODE:

```
android {
    compileSdkVersion 33

    defaultConfig {
        applicationId "com.example.myapp"
        minSdkVersion 21
        targetSdkVersion 33
        versionCode 1
        versionName "1.0"
    }

    signingConfigs {
        release {
```

```
            keyAlias 'my-key'                    // your key alias
            keyPassword 'your-key-password'      // key password
            storeFile file('/path/to/my-release-key.jks') // keystore path
            storePassword 'your-store-password'  // keystore password
        }
    }

    buildTypes {
        release {
            signingConfig signingConfigs.release
            minifyEnabled false
            shrinkResources false
            // Optional ProGuard settings
            proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'
        }
    }
}
```

OUTPUT:

```
C:\Users\saumy>keytool -genkey -v -keystore ~/my-release-key.jks -keyalg RSA -keysize 2048 -validity 10000 -alias my-key

Enter keystore password:
Re-enter new password:
Enter the distinguished name. Provide a single dot (.) to leave a sub-component empty or press ENTER to use the default
value in braces.
What is your first and last name?
What is the name of your organizational unit?
What is the name of your organization?
What is the name of your City or Locality?
What is the name of your State or Province?
What is the two-letter country code for this unit?
Is CN=Saumya Chandwani, OU=Charusat University, O=DEPSTAR, L=Vadodara, ST=Gujarat, C=IN correct?
  [no]:  yes
Generating 2,048 bit RSA key pair and self-signed certificate (SHA384withRSA) with a validity of 10,000 days
        for: CN=Saumya Chandwani, OU=Charusat University, O=DEPSTAR, L=Vadodara, ST=Gujarat, C=IN
[Storing ~/my-release-key.jks]
keytool error: java.io.FileNotFoundException: ~\my-release-key.jks (The system cannot find the path specified)
java.io.FileNotFoundException: ~\my-release-key.jks (The system cannot find the path specified)
        at java.base/java.io.FileOutputStream.open0(Native Method)
        at java.base/java.io.FileOutputStream.open(FileOutputStream.java:289)
        at java.base/java.io.FileOutputStream.<init>(FileOutputStream.java:230)
        at java.base/java.io.FileOutputStream.<init>(FileOutputStream.java:118)
        at java.base/sun.security.tools.keytool.Main.doCommands(Main.java:1385)
        at java.base/sun.security.tools.keytool.Main.run(Main.java:429)
        at java.base/sun.security.tools.keytool.Main.main(Main.java:410)
```

LATEST APPLICATIONS:

Deployed in modern **e-commerce, fintech, and social media apps** for secure login, dashboard, and profile management.Used in **utility, education, and productivity apps** to provide multi-screen navigation with persistent user sessions.

LEARNING OUTCOME:

I know how to sign an APK securely so my Flutter app is trusted, safe from tampering, and ready for release on the Google Play Store or other Android markets.

REFERENCES:

- **Flutter Docs – Signing the app (release mode)** – https://docs.flutter.dev/deployment/android#signing-the-app
- **Medium – Flutter: Generate Signed APK & App Bundle** – https://medium.com/flutter-community/flutter-generate-signed-apk-and-app-bundle-9e45e68ef41a
- **YouTube – Flutter Tutorial: Build & Publish Android App** – https://www.youtube.com/watch?v=KBp_0KYrU9A