Course: BCA

Semester: 6

Prerequisite: Basic Knowledge of Any Programming Language and Object-Oriented concepts.

Course Objective: Introduce with mobile market and mobile application development.

Teaching and Examination Scheme

SEE - Semester End Examination, **T** - Theory, **P** - Practical

Teaching Scheme					Examination Scheme					Total
					Internal Marks			External Marks		
Lecture Hrs/Week	Tutorial Hrs/Week	Lab Hrs/Week	Seminar Hrs/Week	Credit	T	CE	P	T	P	
-	-	2	-	1	-	-	20	-	30	50

Course Outcome

After Learning the Course, the students shall be able to:

1. Students will understand the fundamentals of mobile app development, including Flutter environment setup, widgets, layouts, navigation, and basic app architecture.
2. Students will be able to design and build interactive, responsive user interfaces using Flutter widgets such as Row, Column, Stack, Lists, Forms, and various input components.
3. Students will implement essential app functionalities such as state management, data validation, asynchronous operations, notifications, animations, and background tasks.
4. Students will learn to store and manage data using Shared Preferences, local file handling, and SQLite database with complete CRUD operations.
5. Students will gain the ability to integrate real-world features like WebView, audio playback, maps & location tracking, and build fully functional app modules ready for deployment and testing.

List of Practical

1. **Hello Flutter Emoji App :** Build a simple app showing a big emoji 😄 in the center with a “Tap Me” button that changes the emoji randomly.

Solution :

Features

- Shows a **large emoji** in the center
- “Tap Me” button updates the emoji randomly
- Uses **StatefulWidget**, **setState()**, and basic widgets

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

void main() {
  runApp(const EmojiApp());
}

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      title: 'Emoji App',
      home: const EmojiHome(),
    );
  }
}

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

  @override
  State<EmojiHome> createState() => _EmojiHomeState();
}

class _EmojiHomeState extends State<EmojiHome> {
  final List<String> emojis = [
    '😄', '😂', '😍', '😎', '🥳', '🤩', '😜', '🤗', '😇'
  ];

  String currentEmoji = '😄';
  final Random random = Random();

  void changeEmoji() {
    setState(() {
      currentEmoji = emojis[random.nextInt(emojis.length)];
    });
    }

    @override
    Widget build(BuildContext context) {
      return Scaffold(
        appBar: AppBar(
          title: const Text('Hello Emoji App'),
          centerTitle: true,
        ),
        body: Center(
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
              Text(
                currentEmoji,
                style: const TextStyle(fontSize: 120),
              ),
              const SizedBox(height: 30),
              ElevatedButton(
                onPressed: changeEmoji,
                child: const Text(
                  'Tap Me',
                  style: TextStyle(fontSize: 18),
                ),
              ),
            ],
          ),
        ),
      );
    }
  }
```

2. **Simple Login UI (No Backend)**

Create a login screen using two TextFields.
If both fields are non-empty, show a green "Welcome!" message using SnackBar.
Solution:

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

void main() {
  runApp(const LoginApp());
}

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      title: 'Login Screen',
      home: const LoginScreen(),
   );
  }
}

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

  @override
  State<LoginScreen> createState() => _LoginScreenState();
}

class _LoginScreenState extends State<LoginScreen> {
  final TextEditingController usernameController = TextEditingController();
  final TextEditingController passwordController = TextEditingController();

  void login() {
    if (usernameController.text.isNotEmpty &&
        passwordController.text.isNotEmpty) {
      ScaffoldMessenger.of(context).showSnackBar(
        const SnackBar(
          content: Text('Welcome!'),
          backgroundColor: Colors.green,
        ),
      );
    } else {
      ScaffoldMessenger.of(context).showSnackBar(
        const SnackBar(
          content: Text('Please enter all fields'),
          backgroundColor: Colors.red,
        ),
      );
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Login Screen'),
        centerTitle: true,
      ),
      body: Padding(
        padding: const EdgeInsets.all(20),
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            TextField(
              controller: usernameController,
              decoration: const InputDecoration(
                labelText: 'Username',
                border: OutlineInputBorder(),
              ),
            ),
            const SizedBox(height: 20),
            TextField(
              controller: passwordController,
              obscureText: true,
              decoration: const InputDecoration(
                labelText: 'Password',
                border: OutlineInputBorder(),
              ),
            ),
            const SizedBox(height: 30),
            ElevatedButton(
              onPressed: login,
              child: const Text('Login'),
            ),
          ],
        ),
      ),
    );
  }
}
```

3.

Email Validation

Create a form where the login button only activates when:
Email contains "@"
Password ≥ 6 characters
Show a fun toast like "Great! You typed a real email! 😎"
Solution :

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

void main() {
  runApp(const SmartLoginApp());
}

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

```
 @override
 Widget build(BuildContext context) {
  return const MaterialApp(
   debugShowCheckedModeBanner: false,
   home: LoginForm(),
  );
 }
}

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

 @override
 State<LoginForm> createState() => _LoginFormState();
}

class _LoginFormState extends State<LoginForm> {
 final TextEditingController emailController = TextEditingController();
 final TextEditingController passwordController = TextEditingController();

 bool isButtonEnabled = false;

 void validateForm() {
  setState(() {
   isButtonEnabled =
     emailController.text.contains('@') &&
     passwordController.text.length >= 6;
  });
 }

 void showToast() {
  Fluttertoast.showToast(
   msg: "Great! You typed a real email! 😎",
   toastLength: Toast.LENGTH_SHORT,
   gravity: ToastGravity.BOTTOM,
   backgroundColor: Colors.green,
   textColor: Colors.white,
   fontSize: 16,
  );
 }

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

	```
	    appBar: AppBar(
	      title: const Text("Smart Login Form"),
	      centerTitle: true,
	    ),
	    body: Padding(
	      padding: const EdgeInsets.all(20),
	      child: Column(
	        mainAxisAlignment: MainAxisAlignment.center,
	        children: [
	          TextField(
	            controller: emailController,
	            onChanged: (_) => validateForm(),
	            decoration: const InputDecoration(
	              labelText: 'Email',
	              border: OutlineInputBorder(),
	            ),
	          ),
	          const SizedBox(height: 20),
	          TextField(
	            controller: passwordController,
	            onChanged: (_) => validateForm(),
	            obscureText: true,
	            decoration: const InputDecoration(
	              labelText: 'Password',
	              border: OutlineInputBorder(),
	            ),
	          ),
	          const SizedBox(height: 30),
	          ElevatedButton(
	            onPressed: isButtonEnabled ? showToast : null,
	            child: const Text("Login"),
	          ),
	        ],
	      ),
	    ),
	  );
	 }
	}
	```

4.	**Theme Color Changer : Add 3 Colored Buttons (Blue, Orange, Green).Tapping a button changes the screen background instantly.**
	Solution :
	```
	import 'package:flutter/material.dart';

	void main() {
	  runApp(const ThemeChangerApp());
	}
	```

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

  @override
  Widget build(BuildContext context) {
    return const MaterialApp(
      debugShowCheckedModeBanner: false,
      home: ThemeChangerScreen(),
    );
  }
}

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

  @override
  State<ThemeChangerScreen> createState() => _ThemeChangerScreenState();
}

class _ThemeChangerScreenState extends State<ThemeChangerScreen> {
  Color backgroundColor = Colors.white;

  void changeColor(Color color) {
    setState(() {
      backgroundColor = color;
    });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: backgroundColor,
      appBar: AppBar(
        title: const Text("Theme Color Changer"),
        centerTitle: true,
      ),
      body: Center(
        child: Row(
          mainAxisAlignment: MainAxisAlignment.spaceEvenly,
          children: [
            ElevatedButton(
              style: ElevatedButton.styleFrom(
                backgroundColor: Colors.blue,
              ),
              onPressed: () => changeColor(Colors.blue.shade100),
              child: const Text("Blue"),
            ),
          ElevatedButton(
            style: ElevatedButton.styleFrom(
              backgroundColor: Colors.orange,
            ),
            onPressed: () => changeColor(Colors.orange.shade100),
            child: const Text("Orange"),
          ),
          ElevatedButton(
            style: ElevatedButton.styleFrom(
              backgroundColor: Colors.green,
            ),
            onPressed: () => changeColor(Colors.green.shade100),
            child: const Text("Green"),
          ),
        ],
      ),
    ),
  );
 }
}
```

5. **Counter with Auto Increment**

Create a counter that counts 1…2…3 every second using Timer.

Add buttons:

- Start
- Pause
- Reset

Solution :

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

void main() {
 runApp(const CounterApp());
}

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

 @override
 Widget build(BuildContext context) {
  return const MaterialApp(
   debugShowCheckedModeBanner: false,
   home: CounterScreen(),
  );
 }
}

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

 @override
 State<CounterScreen> createState() => _CounterScreenState();
}

class _CounterScreenState extends State<CounterScreen> {
 int counter = 0;
 Timer? timer;

 void startCounter() {
  timer ??= Timer.periodic(
   const Duration(seconds: 1),
   (Timer t) {
    setState(() {
     counter++;
    });
   },
  );
 }

 void pauseCounter() {
  timer?.cancel();
  timer = null;
 }

 void resetCounter() {
  pauseCounter();
  setState(() {
   counter = 0;
  });
 }

 @override
 void dispose() {
  timer?.cancel();
  super.dispose();
 }

 @override
 Widget build(BuildContext context) {
  return Scaffold(
   appBar: AppBar(
    title: const Text("Auto Increment Counter"),
    centerTitle: true,
   ),
   body: Column(
    mainAxisAlignment: MainAxisAlignment.center,
    children: [
          Text(
            counter.toString(),
            style: const TextStyle(
              fontSize: 80,
              fontWeight: FontWeight.bold,
            ),
          ),
          const SizedBox(height: 40),
          Row(
            mainAxisAlignment: MainAxisAlignment.spaceEvenly,
            children: [
              ElevatedButton(
                onPressed: startCounter,
                child: const Text("Start"),
              ),
              ElevatedButton(
                onPressed: pauseCounter,
                child: const Text("Pause"),
              ),
              ElevatedButton(
                onPressed: resetCounter,
                child: const Text("Reset"),
              ),
            ],
          ),
        ],
      ),
    );
  }
}
```

6.

Loading Screen (Progress Bar)

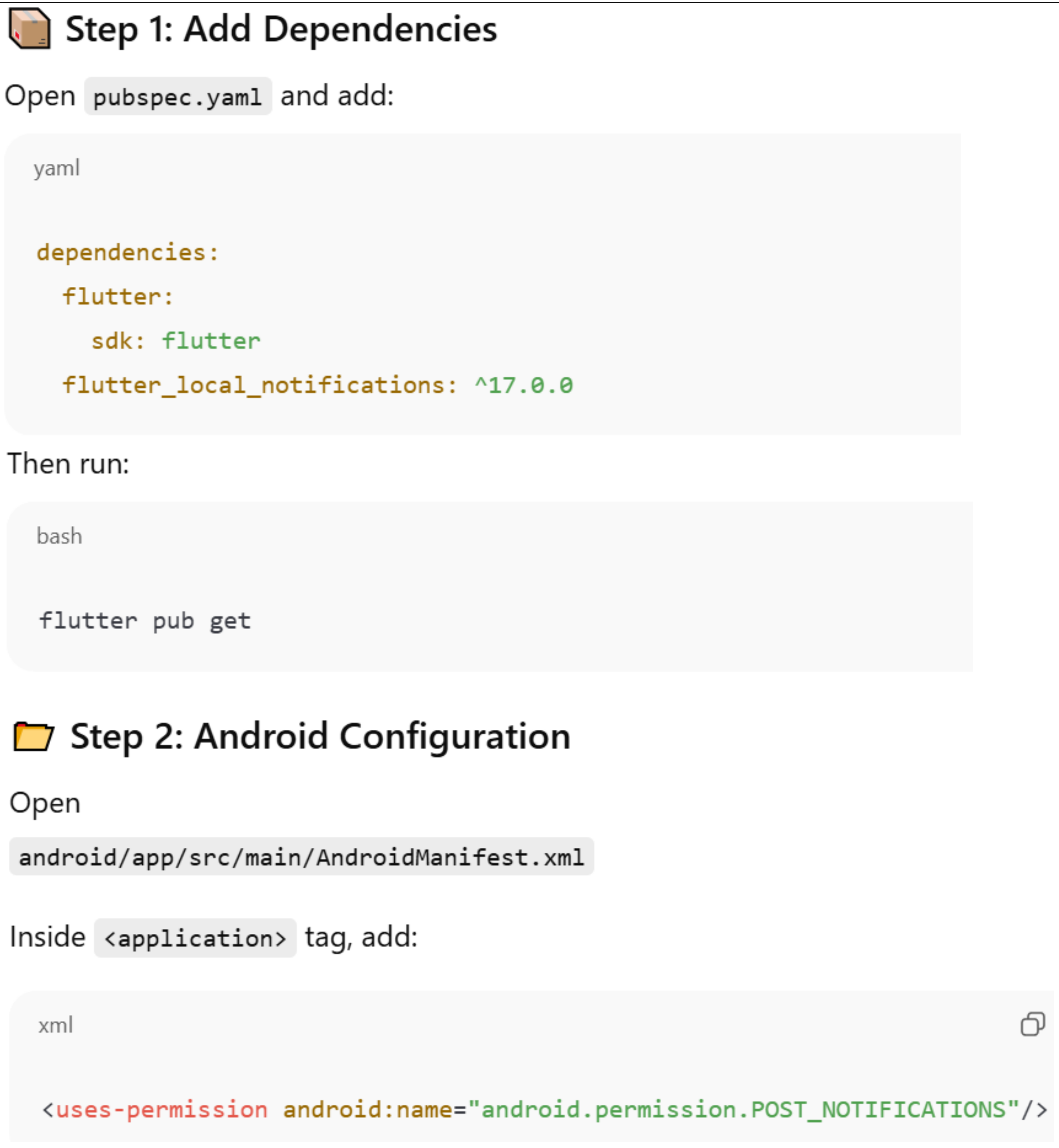
On button click:

- Start progress from 0% to 100%
- Show “Loading your awesome content…”
- After done, show “Finished! 🎉”

Solution :

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

void main() {
  runApp(const LoadingApp());
}

class LoadingApp extends StatelessWidget {
  const LoadingApp({super.key});
  @override
  Widget build(BuildContext context) {
    return const MaterialApp(
      debugShowCheckedModeBanner: false,
      home: LoadingScreen(),
    );
  }
}

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

  @override
  State<LoadingScreen> createState() => _LoadingScreenState();
}

class _LoadingScreenState extends State<LoadingScreen> {
  double progress = 0.0;
  String message = "";
  Timer? timer;

  void startLoading() {
    setState(() {
      progress = 0.0;
      message = "Loading your awesome content...";
    });

    timer?.cancel();
    timer = Timer.periodic(const Duration(milliseconds: 100), (Timer t) {
      setState(() {
        progress += 0.01;
        if (progress >= 1.0) {
          progress = 1.0;
          message = "Finished! 🎉";
          t.cancel();
        }
      });
    });
  }

  @override
  void dispose() {
    timer?.cancel();
    super.dispose();
  }
```

	```
	  @override
	  Widget build(BuildContext context) {
	    return Scaffold(
	      appBar: AppBar(
	        title: const Text("Loading Screen"),
	        centerTitle: true,
	      ),
	      body: Padding(
	        padding: const EdgeInsets.all(20),
	        child: Column(
	          mainAxisAlignment: MainAxisAlignment.center,
	          children: [
	            LinearProgressIndicator(
	              value: progress,
	              minHeight: 20,
	              color: Colors.blue,
	              backgroundColor: Colors.grey.shade300,
	            ),
	            const SizedBox(height: 20),
	            Text(
	              message,
	              style: const TextStyle(fontSize: 18),
	            ),
	            const SizedBox(height: 40),
	            ElevatedButton(
	              onPressed: startLoading,
	              child: const Text("Start Loading"),
	            ),
	          ],
	        ),
	      ),
	    );
	  }
	}
	```
7.	**Reminder App (Simple Notifications)** Let the user enter a message. After 5 seconds, show a notification with that message. Solution :



📦 Step 1: Add Dependencies

Open `pubspec.yaml` and add:

yaml

```yaml
dependencies:
  flutter:
    sdk: flutter
  flutter_local_notifications: ^17.0.0
```

Then run:

bash

```bash
flutter pub get
```

📁 Step 2: Android Configuration

Open

`android/app/src/main/AndroidManifest.xml`

Inside `<application>` tag, add:

xml

```xml
<uses-permission android:name="android.permission.POST_NOTIFICATIONS"/>
```

Main. Dart

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

void main() {
  runApp(const ReminderApp());
}

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      title: 'Reminder App',
      home: const ReminderScreen(),
    );
  }
}

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

  @override
  State<ReminderScreen> createState() => _ReminderScreenState();
}

class _ReminderScreenState extends State<ReminderScreen> {
  final TextEditingController _controller = TextEditingController();

  FlutterLocalNotificationsPlugin notificationsPlugin =
      FlutterLocalNotificationsPlugin();

  @override
  void initState() {
    super.initState();
    initializeNotification();
  }

  void initializeNotification() async {
    const AndroidInitializationSettings androidSettings =
        AndroidInitializationSettings('@mipmap/ic_launcher');

    const InitializationSettings settings =
        InitializationSettings(android: androidSettings);

    await notificationsPlugin.initialize(settings);
  }

  void showNotification(String message) async {
    const AndroidNotificationDetails androidDetails =
        AndroidNotificationDetails(
      'reminder_channel',
      'Reminders',
      importance: Importance.max,
      priority: Priority.high,
    );

    const NotificationDetails notificationDetails =
        NotificationDetails(android: androidDetails);

    await notificationsPlugin.show(
      0,
      'Reminder ⏰',
    message,
    notificationDetails,
  );
}

void setReminder() {
  String message = _controller.text;

  if (message.isEmpty) return;

  Future.delayed(const Duration(seconds: 5), () {
    showNotification(message);
  });

  ScaffoldMessenger.of(context).showSnackBar(
    const SnackBar(content: Text('Reminder set for 5 seconds')),
  );

  _controller.clear();
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: const Text('Simple Reminder App'),
      backgroundColor: Colors.blue,
    ),
    body: Padding(
      padding: const EdgeInsets.all(16),
      child: Column(
        children: [
          TextField(
            controller: _controller,
            decoration: const InputDecoration(
              labelText: 'Enter reminder message',
              border: OutlineInputBorder(),
            ),
          ),
          const SizedBox(height: 20),
          ElevatedButton(
            onPressed: setReminder,
            child: const Text('Set Reminder'),
          ),
        ],
      ),
    ),
  );
  }
}
```

8. **Prime Number Finder : User enters a lower & upper range.App finds all prime numbers (async) and displays them.After calculation completes → send a notification.**

Solution [main.dart] :

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

void main() {
  runApp(const PrimeApp());
}

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      home: const PrimeScreen(),
    );
  }
}

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

  @override
  State<PrimeScreen> createState() => _PrimeScreenState();
}

class _PrimeScreenState extends State<PrimeScreen> {
  final TextEditingController lowerController = TextEditingController();
  final TextEditingController upperController = TextEditingController();

  List<int> primes = [];
  bool isLoading = false;

  FlutterLocalNotificationsPlugin notificationsPlugin =
      FlutterLocalNotificationsPlugin();

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

```
void initializeNotification() async {
  const AndroidInitializationSettings androidSettings =
      AndroidInitializationSettings('@mipmap/ic_launcher');

  const InitializationSettings settings =
      InitializationSettings(android: androidSettings);

  await notificationsPlugin.initialize(settings);
}

Future<void> showNotification() async {
  const AndroidNotificationDetails androidDetails =
      AndroidNotificationDetails(
    'prime_channel',
    'Prime Finder',
    importance: Importance.max,
    priority: Priority.high,
  );

  const NotificationDetails details =
      NotificationDetails(android: androidDetails);

  await notificationsPlugin.show(
    0,
    'Prime Calculation Done ✅',
    'Prime numbers are ready to view',
    details,
  );
}

bool isPrime(int n) {
  if (n <= 1) return false;
  for (int i = 2; i <= n ~/ 2; i++) {
    if (n % i == 0) return false;
  }
  return true;
}

Future<void> findPrimesAsync(int low, int high) async {
  setState(() {
    isLoading = true;
    primes.clear();
  });

  await Future.delayed(const Duration(seconds: 1)); // simulate async work

  List<int> result = [];
  for (int i = low; i <= high; i++) {
    if (isPrime(i)) {
      result.add(i);
    }
  }

  setState(() {
    primes = result;
    isLoading = false;
  });

  showNotification();
}

void startCalculation() {
  int low = int.parse(lowerController.text);
  int high = int.parse(upperController.text);
  findPrimesAsync(low, high);
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: const Text('Prime Number Finder'),
      backgroundColor: Colors.green,
    ),
    body: Padding(
      padding: const EdgeInsets.all(16),
      child: Column(
        children: [
          TextField(
            controller: lowerController,
            keyboardType: TextInputType.number,
            decoration: const InputDecoration(
              labelText: 'Lower Range',
              border: OutlineInputBorder(),
            ),
          ),
          const SizedBox(height: 10),
          TextField(
            controller: upperController,
            keyboardType: TextInputType.number,
            decoration: const InputDecoration(
              labelText: 'Upper Range',
              border: OutlineInputBorder(),
            ),
          ),
```

	```
	          const SizedBox(height: 20),
	          ElevatedButton(
	            onPressed: startCalculation,
	            child: const Text('Find Primes'),
	          ),
	          const SizedBox(height: 20),
	          if (isLoading)
	            const CircularProgressIndicator()
	          else
	            Expanded(
	              child: ListView.builder(
	                itemCount: primes.length,
	                itemBuilder: (context, index) {
	                  return ListTile(
	                    title: Text(primes[index].toString()),
	                  );
	                },
	              ),
	            ),
	        ],
	      ),
	    ),
	  );
	 }
	}
	```

9.	**Contacts Viewer**
	
	• **Option A (Easy):**
	Instead of phone contacts, use a simple list of name–number pairs. (no permissions).
	• **Option B (Advanced):**
	Actually fetch phone contacts using contacts_service.
	
	Display them using ListView with icons.
	Solution [main.dart]
	```
	import 'package:flutter/material.dart';

	void main() {
	  runApp(const ContactsApp());
	}

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

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

	```
	    return MaterialApp(
	      debugShowCheckedModeBanner: false,
	      home: const ContactsScreen(),
	    );
	  }
	}

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

	  final List<Map<String, String>> contacts = const [
	    {'name': 'Ravi Patel', 'number': '9876543210'},
	    {'name': 'Neha Shah', 'number': '9123456789'},
	    {'name': 'Amit Kumar', 'number': '9988776655'},
	    {'name': 'Priya Mehta', 'number': '9012345678'},
	  ];

	  @override
	  Widget build(BuildContext context) {
	    return Scaffold(
	      appBar: AppBar(
	        title: const Text('Contacts Viewer'),
	        backgroundColor: Colors.blue,
	      ),
	      body: ListView.builder(
	        itemCount: contacts.length,
	        itemBuilder: (context, index) {
	          return ListTile(
	            leading: const Icon(Icons.person),
	            title: Text(contacts[index]['name']!),
	            subtitle: Text(contacts[index]['number']!),
	          );
	        },
	      ),
	    );
	  }
	}
	```

10.	**Dynamic Cards Generator :** User enters a number (e.g., 5).Generate 5 colorful cards on the next page with dummy titles like “Card 1”, “Card 2”, etc.
	Solution [Main.dart]
	```
	import 'dart:math';
	import 'package:flutter/material.dart';
	```

```
void main() {
  runApp(const CardApp());
}

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      home: const InputScreen(),
    );
  }
}

// ---------- First Screen ----------
class InputScreen extends StatefulWidget {
  const InputScreen({super.key});

  @override
  State<InputScreen> createState() => _InputScreenState();
}

class _InputScreenState extends State<InputScreen> {
  final TextEditingController controller = TextEditingController();

  void goToCardsPage() {
    int count = int.parse(controller.text);

    Navigator.push(
      context,
      MaterialPageRoute(
        builder: (context) => CardsScreen(cardCount: count),
      ),
    );
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Dynamic Cards Generator'),
        backgroundColor: Colors.blue,
      ),
      body: Padding(
        padding: const EdgeInsets.all(16),
        child: Column(
          children: [
            TextField(
              controller: controller,
              keyboardType: TextInputType.number,
              decoration: const InputDecoration(
                labelText: 'Enter number of cards',
                border: OutlineInputBorder(),
              ),
            ),
            const SizedBox(height: 20),
            ElevatedButton(
              onPressed: goToCardsPage,
              child: const Text('Generate Cards'),
            ),
          ],
        ),
      ),
    );
  }
}

// ---------- Second Screen ----------
class CardsScreen extends StatelessWidget {
  final int cardCount;
  CardsScreen({super.key, required this.cardCount});

  final Random random = Random();

  Color getRandomColor() {
    return Color.fromARGB(
      255,
      random.nextInt(256),
      random.nextInt(256),
      random.nextInt(256),
    );
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Generated Cards'),
        backgroundColor: Colors.green,
      ),
```

	```
	    body: ListView.builder(
	     itemCount: cardCount,
	     itemBuilder: (context, index) {
	      return Card(
	       color: getRandomColor(),
	       margin: const EdgeInsets.all(10),
	       elevation: 6,
	       child: ListTile(
	        title: Text(
	         'Card ${index + 1}',
	         style: const TextStyle(
	          color: Colors.white,
	          fontSize: 18,
	         ),
	        ),
	       ),
	      );
	     },
	    ),
	   );
	  }
	}
	```
11.	**Fake Dialer :User enters a phone number → tap “Call” → show a dialog:📞 Calling 9876543210…** **Solution [main.dart]**

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

void main() {
 runApp(const FakeDialerApp());
}

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

 @override
 Widget build(BuildContext context) {
  return MaterialApp(
   debugShowCheckedModeBanner: false,
   home: const DialerScreen(),
  );
 }
}

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

 @override
 State<DialerScreen> createState() => _DialerScreenState();
}

class _DialerScreenState extends State<DialerScreen> {
 final TextEditingController phoneController = TextEditingController();
```

```
 void showCallDialog() {
   String number = phoneController.text;

   if (number.isEmpty) return;

   showDialog(
     context: context,
     builder: (context) => AlertDialog(
       title: const Text('Calling'),
       content: Text('📞 Calling $number...'),
       actions: [
         TextButton(
           onPressed: () => Navigator.pop(context),
           child: const Text('End Call'),
         ),
       ],
     ),
   );
 }

 @override
 Widget build(BuildContext context) {
   return Scaffold(
     appBar: AppBar(
       title: const Text('Fake Dialer'),
       backgroundColor: Colors.green,
     ),
     body: Padding(
       padding: const EdgeInsets.all(16),
       child: Column(
         children: [
           TextField(
             controller: phoneController,
             keyboardType: TextInputType.phone,
             decoration: const InputDecoration(
               labelText: 'Enter phone number',
               border: OutlineInputBorder(),
               prefixIcon: Icon(Icons.phone),
             ),
           ),
           const SizedBox(height: 20),
           ElevatedButton.icon(
             onPressed: showCallDialog,
             icon: const Icon(Icons.call),
             label: const Text('Call'),
           ),
         ],
       ),
     ),
   );
 }
}
```

12.

Basic Messaging Mock App : A simple messaging simulation:

- User types a message
- Adds it to a list displayed like chat bubbles
- Buttons: Send, Clear All

Solution [main.dart]

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

void main() {
  runApp(const MessagingApp());
}

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      home: const ChatScreen(),
    );
  }
}

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

  @override
  State<ChatScreen> createState() => _ChatScreenState();
}

class _ChatScreenState extends State<ChatScreen> {
  final TextEditingController messageController = TextEditingController();
  final List<String> messages = [];

  void sendMessage() {
    if (messageController.text.isEmpty) return;

    setState(() {
      messages.add(messageController.text);
    });

    messageController.clear();
  }

  void clearMessages() {
    setState(() {
      messages.clear();
    });
```

```
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: const Text('Messaging Mock App'),
      backgroundColor: Colors.teal,
      actions: [
        IconButton(
          onPressed: clearMessages,
          icon: const Icon(Icons.delete),
        ),
      ],
    ),
    body: Column(
      children: [
        Expanded(
          child: ListView.builder(
            itemCount: messages.length,
            itemBuilder: (context, index) {
              return Align(
                alignment: Alignment.centerRight,
                child: Container(
                  margin: const EdgeInsets.all(8),
                  padding: const EdgeInsets.all(12),
                  decoration: BoxDecoration(
                    color: Colors.teal.shade300,
                    borderRadius: BorderRadius.circular(15),
                  ),
                  child: Text(
                    messages[index],
                    style: const TextStyle(color: Colors.white),
                  ),
                ),
              );
            },
          ),
        ),
        Padding(
          padding: const EdgeInsets.all(8),
          child: Row(
            children: [
              Expanded(
                child: TextField(
                  controller: messageController,
                  decoration: const InputDecoration(
                    hintText: 'Type a message...',
                      border: OutlineInputBorder(),
                    ),
                  ),
                ),
                const SizedBox(width: 8),
                ElevatedButton(
                  onPressed: sendMessage,
                  child: const Text('Send'),
                ),
              ],
            ),
          ),
        ],
      ),
    );
  }
}
```

13.

File Save (Text File) – Notes App

Create a small notes app:

- Enter text
- Tap “Save Note”
- Save to local file
- Tap “Read Note” to display saved content

Solution [Main.dart]

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

void main() {
  runApp(const NotesApp());
}

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      home: const NotesScreen(),
    );
  }
}

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

```
 @override
 State<NotesScreen> createState() => _NotesScreenState();
}

class _NotesScreenState extends State<NotesScreen> {
 final TextEditingController noteController = TextEditingController();
 String savedNote = '';

 Future<String> getFilePath() async {
  final directory = await getApplicationDocumentsDirectory();
  return '${directory.path}/note.txt';
 }

 Future<void> saveNote() async {
  final path = await getFilePath();
  final file = File(path);

  await file.writeAsString(noteController.text);

  ScaffoldMessenger.of(context).showSnackBar(
   const SnackBar(content: Text('Note Saved Successfully')),
  );

  noteController.clear();
 }

 Future<void> readNote() async {
  try {
   final path = await getFilePath();
   final file = File(path);

   String content = await file.readAsString();

   setState(() {
    savedNote = content;
   });
  } catch (e) {
   setState(() {
    savedNote = 'No saved note found';
   });
  }
 }

 @override
 Widget build(BuildContext context) {
  return Scaffold(
   appBar: AppBar(
    title: const Text('Notes App'),
    backgroundColor: Colors.blueGrey,
      ),
      body: Padding(
        padding: const EdgeInsets.all(16),
        child: Column(
          children: [
            TextField(
              controller: noteController,
              maxLines: 4,
              decoration: const InputDecoration(
                labelText: 'Enter your note',
                border: OutlineInputBorder(),
              ),
            ),
            const SizedBox(height: 20),
            Row(
              mainAxisAlignment: MainAxisAlignment.spaceEvenly,
              children: [
                ElevatedButton(
                  onPressed: saveNote,
                  child: const Text('Save Note'),
                ),
                ElevatedButton(
                  onPressed: readNote,
                  child: const Text('Read Note'),
                ),
              ],
            ),
            const SizedBox(height: 20),
            const Text(
              'Saved Note:',
              style: TextStyle(fontWeight: FontWeight.bold),
            ),
            const SizedBox(height: 10),
            Expanded(
              child: SingleChildScrollView(
                child: Text(savedNote),
              ),
            ),
          ],
        ),
      ),
    );
  }
}
```

14.

Dark Mode Toggle (Shared Preferences)

Add a Switch: ON = Dark Mode, OFF = Light Mode.

Save the user’s choice so the app remembers it next time.

Solution [Main.dart]

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

void main() {
 runApp(const MyApp());
}

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

 @override
 State<MyApp> createState() => _MyAppState();
}

class _MyAppState extends State<MyApp> {
 bool isDarkMode = false;

 @override
 void initState() {
  super.initState();
  loadTheme();
 }

 // Load saved theme
 Future<void> loadTheme() async {
  final prefs = await SharedPreferences.getInstance();
  setState(() {
   isDarkMode = prefs.getBool('darkMode') ?? false;
  });
 }

 // Save theme preference
 Future<void> saveTheme(bool value) async {
  final prefs = await SharedPreferences.getInstance();
  prefs.setBool('darkMode', value);
 }

 @override
 Widget build(BuildContext context) {
  return MaterialApp(
   debugShowCheckedModeBanner: false,
   theme: isDarkMode ? ThemeData.dark() : ThemeData.light(),
   home: HomeScreen(
    isDarkMode: isDarkMode,
    onThemeChanged: (value) {
          setState(() {
            isDarkMode = value;
          });
          saveTheme(value);
        },
      ),
    );
  }
}

class HomeScreen extends StatelessWidget {
  final bool isDarkMode;
  final Function(bool) onThemeChanged;

  const HomeScreen({
    super.key,
    required this.isDarkMode,
    required this.onThemeChanged,
  });

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Dark Mode Toggle'),
      ),
      body: Center(
        child: Row(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            const Text('Light'),
            Switch(
              value: isDarkMode,
              onChanged: onThemeChanged,
            ),
            const Text('Dark'),
          ],
        ),
      ),
    );
  }
}
```

15. **Mini Student Records (SQLite)**

Using sqflite:

- Add student (id, name, address, contact)
- Update
- Delete
- List all students

Use Cards + ListTiles for clean UI.

Solution [main.dart]

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

void main() {
  runApp(const StudentApp());
}

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

  @override
  Widget build(BuildContext context) {
    return const MaterialApp(
      debugShowCheckedModeBanner: false,
      home: StudentScreen(),
    );
  }
}

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

  @override
  State<StudentScreen> createState() => _StudentScreenState();
}

class _StudentScreenState extends State<StudentScreen> {
  Database? database;
  List<Map<String, dynamic>> students = [];

  final idController = TextEditingController();
  final nameController = TextEditingController();
  final addressController = TextEditingController();
  final contactController = TextEditingController();

  @override
```

```
void initState() {
  super.initState();
  initDB();
}

Future<void> initDB() async {
  database = await openDatabase(
    join(await getDatabasesPath(), 'students.db'),
    version: 1,
    onCreate: (db, version) async {
      await db.execute(
        '''CREATE TABLE students(
          id INTEGER PRIMARY KEY,
          name TEXT,
          address TEXT,
          contact TEXT
        )''',
      );
    },
  );
  fetchStudents();
}

Future<void> fetchStudents() async {
  final data = await database!.query('students');
  setState(() {
    students = data;
  });
}

Future<void> addStudent() async {
  await database!.insert('students', {
    'id': int.parse(idController.text),
    'name': nameController.text,
    'address': addressController.text,
    'contact': contactController.text,
  });
  clearFields();
  fetchStudents();
}

Future<void> updateStudent() async {
  await database!.update(
    'students',
    {
     'name': nameController.text,
     'address': addressController.text,
     'contact': contactController.text,
    },
    where: 'id = ?',
    whereArgs: [int.parse(idController.text)],
  );
  clearFields();
  fetchStudents();
}

Future<void> deleteStudent(int id) async {
  await database!.delete(
    'students',
    where: 'id = ?',
    whereArgs: [id],
  );
  fetchStudents();
}

void clearFields() {
  idController.clear();
  nameController.clear();
  addressController.clear();
  contactController.clear();
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: const Text('Student Records (SQLite)'),
      backgroundColor: Colors.indigo,
    ),
    body: Padding(
      padding: const EdgeInsets.all(10),
      child: Column(
        children: [
          TextField(
            controller: idController,
            keyboardType: TextInputType.number,
            decoration: const InputDecoration(labelText: 'Student ID'),
          ),
          TextField(
            controller: nameController,
            decoration: const InputDecoration(labelText: 'Name'),
          ),
          TextField(
            controller: addressController,
            decoration: const InputDecoration(labelText: 'Address'),
          ),
          TextField(
            controller: contactController,
            decoration: const InputDecoration(labelText: 'Contact'),
          ),
          const SizedBox(height: 10),
          Row(
            mainAxisAlignment: MainAxisAlignment.spaceEvenly,
            children: [
              ElevatedButton(
                onPressed: addStudent,
                child: const Text('Add'),
              ),
              ElevatedButton(
                onPressed: updateStudent,
                child: const Text('Update'),
              ),
            ],
          ),
          const SizedBox(height: 10),
          Expanded(
            child: ListView.builder(
              itemCount: students.length,
              itemBuilder: (context, index) {
                final student = students[index];
                return Card(
                  elevation: 4,
                  margin: const EdgeInsets.all(6),
                  child: ListTile(
                    leading: const Icon(Icons.school),
                    title: Text(student['name']),
                    subtitle: Text(
                      'ID: ${student['id']}\n'
                      'Address: ${student['address']}\n'
                      'Contact: ${student['contact']}',
                    ),
                    trailing: IconButton(
                      icon: const Icon(Icons.delete, color: Colors.red),
                      onPressed: () =>
                          deleteStudent(student['id']),
```

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