CS506PC: UI DESIGN - FLUTTER

B.Tech III Year I Semester - Lab Manual

Includes Aim, Procedure, Sample Code, Expected Output, and Viva Questions for each experiment.

Experiment 1: Install Flutter & Dart, Simple Dart Program

Aim: Install Flutter and Dart SDK and run a simple Dart program.

Procedure:

1. Install Flutter SDK from https://flutter.dev and configure PATH.
2. Run `flutter doctor` to verify dependencies.
3. Create a new Flutter project: `flutter create lab1`.
4. Open lib/main.dart and run the app on an emulator or device.

Sample Dart Program (command-line):

```
// Save as bin/main.dart
void main() {
  print('Hello from Dart!');
}
```

Sample Flutter App (main.dart):

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

void main() => runApp(const MyApp());

class MyApp extends StatelessWidget {
  const MyApp({super.key});
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(title: const Text('Experiment 1')),
        body: const Center(child: Text('Hello from Flutter!')),
      ),
    );
  }
}
```

Expected Output: App displays 'Hello from Flutter!' in the center.

Viva Questions: What does `flutter doctor` check? How do you run a Dart script?

Experiment 2: Explore Widgets & Layouts

Aim: Learn common widgets and layout mechanisms (Row, Column, Stack).

Sample Code:

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

void main() => runApp(const WidgetDemo());

class WidgetDemo extends StatelessWidget {
  const WidgetDemo({super.key});
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(title: const Text('Widgets & Layouts')),
        body: Column(
          children: [
            const Text('Top Text'),
            Row(
              mainAxisAlignment: MainAxisAlignment.spaceAround,
              children: const [
                Icon(Icons.thumb_up),
                Icon(Icons.favorite),
                Icon(Icons.share),
              ],
            ),
            Expanded(
              child: Stack(
                children: [
                  Container(color: Colors.blueAccent),
                  const Positioned(
                    left: 20, top: 20,
                    child: Text('Stacked Text', style: TextStyle(color: Colors.white)),
                  ),
                ],
              ),
            ),
          ],
        ),
      ),
    );
  }
}
```

Expected Output: A Column with text, icons in a Row, and a stacked container.

Viva Questions: When to use Stack vs Column? What's Expanded used for?

Experiment 3: Responsive UI (MediaQuery & LayoutBuilder)

Aim: Build UI that adapts to screen sizes.

Sample Code:

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

void main() => runApp(const ResponsiveApp());

class ResponsiveApp extends StatelessWidget {
  const ResponsiveApp({super.key});
  @override
  Widget build(BuildContext context) {
    return MaterialApp(home: const HomePage());
  }
}

class HomePage extends StatelessWidget {
  const HomePage({super.key});
  @override
  Widget build(BuildContext context) {
    final width = MediaQuery.of(context).size.width;
    final isWide = width > 600;
    return Scaffold(
      appBar: AppBar(title: const Text('Responsive UI')),
      body: isWide ? _wideLayout() : _narrowLayout(),
    );
  }

  Widget _wideLayout() {
    return Row(
      children: const [
        Expanded(child: Center(child: Text('Left Pane'))),
        Expanded(child: Center(child: Text('Right Pane'))),
      ],
    );
  }

  Widget _narrowLayout() {
    return ListView(
      children: const [
        ListTile(title: Text('Item 1')),
        ListTile(title: Text('Item 2')),
      ],
    );
  }
}
```

Expected Output: Two-column layout on wide screens; single-column list on narrow screens.

Experiment 4: Navigation & Named Routes

Sample Code:

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

void main() => runApp(const NavApp());

class NavApp extends StatelessWidget {
  const NavApp({super.key});
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      initialRoute: '/',
      routes: {
        '/': (context) => const HomeScreen(),
        '/second': (context) => const SecondScreen(),
      },
    );
  }
}

class HomeScreen extends StatelessWidget {
  const HomeScreen({super.key});
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('Home')),
      body: Center(
        child: ElevatedButton(
          onPressed: () => Navigator.pushNamed(context, '/second'),
          child: const Text('Go to Second'),
        ),
      ),
    );
  }
}

class SecondScreen extends StatelessWidget {
  const SecondScreen({super.key});
  @override
  Widget build(BuildContext context) {
    return Scaffold(appBar: AppBar(title: const Text('Second')), body: const Center(child: Text('Second Screen')));
  }
}
```

Expected Output: Button navigates to second screen using named route.

Experiment 5: Stateful vs Stateless & State Management

Aim: Understand stateful/stateless widgets and basic state management using setState and Provider.

Sample Code (setState counter):

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

void main() => runApp(const CounterApp());

class CounterApp extends StatelessWidget {
  const CounterApp({super.key});
  @override
  Widget build(BuildContext context) {
    return MaterialApp(home: const CounterPage());
  }
}

class CounterPage extends StatefulWidget {
  const CounterPage({super.key});
  @override
  State<CounterPage> createState() => _CounterPageState();
}

class _CounterPageState extends State<CounterPage> {
  int count = 0;
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('Counter')),
      body: Center(child: Text('Count: $count')),
      floatingActionButton: FloatingActionButton(
        onPressed: () => setState(() => count++),
        child: const Icon(Icons.add),
      ),
    );
  }
}
```

Note on Provider: To use Provider, add `provider` to pubspec.yaml and create a ChangeNotifier class to hold state.

Experiment 6: Custom Widgets, Themes & Styles

Sample Code:

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

void main() => runApp(const ThemedApp());

class ThemedApp extends StatelessWidget {
  const ThemedApp({super.key});
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      theme: ThemeData(
        primarySwatch: Colors.teal,
        textTheme: const TextTheme(bodyMedium: TextStyle(fontSize: 16)),
      ),
      home: const Home(),
    );
  }
}

class CustomCard extends StatelessWidget {
  final String title;
  const CustomCard(this.title, {super.key});
  @override
  Widget build(BuildContext context) {
    return Card(margin: const EdgeInsets.all(8), child: Padding(padding: const EdgeInsets.all(16), child: Text(title)));
  }
}

class Home extends StatelessWidget {
  const Home({super.key});
  @override
  Widget build(BuildContext context) {
    return Scaffold(appBar: AppBar(title: const Text('Themes & Custom Widgets')), body: ListView(children: const [
      CustomCard('Card 1'),
      CustomCard('Card 2'),
    ]));
  }
}
```

Experiment 7: Forms, Validation & Error Handling

Sample Code:

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

void main() => runApp(const FormApp());

class FormApp extends StatelessWidget {
  const FormApp({super.key});
  @override
  Widget build(BuildContext context) => MaterialApp(home: const FormPage());
}

class FormPage extends StatefulWidget {
  const FormPage({super.key});
  @override
  State<FormPage> createState() => _FormPageState();
}

class _FormPageState extends State<FormPage> {
  final _formKey = GlobalKey<FormState>();
  String name = '';
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('Form Validation')),
      body: Padding(
        padding: const EdgeInsets.all(16.0),
        child: Form(
          key: _formKey,
          child: Column(children: [
            TextFormField(
              decoration: const InputDecoration(labelText: 'Name'),
              validator: (value) => (value==null || value.isEmpty) ? 'Enter name' : null,
              onSaved: (v) => name = v ?? '',
            ),
            const SizedBox(height: 16),
            ElevatedButton(
              child: const Text('Submit'),
              onPressed: () {
                if (_formKey.currentState?.validate() ?? false) {
                  _formKey.currentState?.save();
                  ScaffoldMessenger.of(context).showSnackBar(SnackBar(content: Text('Hello, $name')));
                }
              },
            ),
          ]),
        ),
      ),
    );
  }
}
```

Experiment 8: Animations (Fade & Slide)

Sample Code (FadeTransition):

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

void main() => runApp(const AnimApp());

class AnimApp extends StatelessWidget {
  const AnimApp({super.key});
  @override
  Widget build(BuildContext context) => MaterialApp(home: const AnimPage());
}

class AnimPage extends StatefulWidget {
  const AnimPage({super.key});
  @override
  State<AnimPage> createState() => _AnimPageState();
}

class _AnimPageState extends State<AnimPage> with SingleTickerProviderStateMixin {
  late final AnimationController _controller;
  late final Animation<double> _animation;
  @override
  void initState() {
    super.initState();
    _controller = AnimationController(vsync: this, duration: const Duration(seconds: 2))..repeat(reverse: true);
    _animation = CurvedAnimation(parent: _controller, curve: Curves.easeInOut);
  }
  @override
  void dispose() {
    _controller.dispose();
    super.dispose();
  }
  @override
  Widget build(BuildContext context) {
    return Scaffold(appBar: AppBar(title: const Text('Animation')), body: Center(child: FadeTransition(opacity: _animat
  }
}
```

Experiment 9: Fetch Data from REST API

Note: Add `http: ^0.13.0` (or latest) to pubspec.yaml.

Sample Code:

```
import 'package:flutter/material.dart';
import 'package:http/http.dart' as http;
import 'dart:convert';

void main() => runApp(const ApiApp());

class ApiApp extends StatelessWidget {
  const ApiApp({super.key});
  @override
  Widget build(BuildContext context) => MaterialApp(home: const ApiPage());
}

class ApiPage extends StatefulWidget {
  const ApiPage({super.key});
  @override
  State<ApiPage> createState() => _ApiPageState();
}

class _ApiPageState extends State<ApiPage> {
  List posts = [];
  bool loading = true;

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

  Future<void> fetchPosts() async {
    final res = await http.get(Uri.parse('https://jsonplaceholder.typicode.com/posts'));
    if (res.statusCode == 200) {
      setState(() {
        posts = json.decode(res.body);
        loading = false;
      });
    } else {
      setState(() => loading = false);
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('REST API')),
      body: loading ? const Center(child: CircularProgressIndicator()) : ListView.builder(
        itemCount: posts.length,
        itemBuilder: (context, i) => ListTile(title: Text(posts[i]['title'])),
      ),
    );
  }
}
```

Experiment 10: Unit Tests & Debugging

Sample Unit Test:

```
// In test/widget_test.dart
import 'package:flutter_test/flutter_test.dart';
import 'package:flutter/material.dart';
import 'package:your_app/main.dart';

void main() {
  testWidgets('App shows Hello from Flutter', (WidgetTester tester) async {
    await tester.pumpWidget(const MyApp());
    expect(find.text('Hello from Flutter!'), findsOneWidget);
  });
}
```

Debugging Tools: Use `flutter run --verbose`, DevTools for widget inspector, performance and network.

End of Lab Manual