Flutter Internal Exam (Executable Version)

Contains: 10 Questions (A & B) — Short theory + Full runnable Flutter code for each part

Question 1. A) Explain about Installation process of Flutter and Dart SDK.

- Install Flutter SDK from flutter.dev - Add Flutter to PATH - Run flutter doctor - Dart SDK comes with Flutter - Install Android Studio / VS Code Flutter plugin

```
// Question 1B: Debugging example (executable Dart console program)
void main() {
  int a = 10, b = 0;
  try {
    print(a ~/ b);
  } catch (e) {
    print('Error caught: $e');
  }
}
```

B) Write flutter code by using flutter debugging tools to identify and fix issues.

- Use try-catch for runtime errors - Use Flutter DevTools for inspecting widgets, performance, and logs - Use breakpoints and logging (print/debugPrint)

```
// Question 1B alternative (simple Flutter app demonstrating error handling)
import 'package:flutter/material.dart';
void main() => runApp(DebugApp());

class DebugApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(home: Scaffold(body: Center(child: DebugWidget())));
  }
}

class DebugWidget extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    try {
      final int a = 5;
      final int b = 0;
      final int c = a ~/ b; // will throw
      return Text('Result: $c');
    } catch (e) {
      return Text('Caught error: $e');
    }
  }
}
```

Question 2. A) Create User Interface (UI) with Text widgets in Flutter.

Text widget displays static or dynamic content in UI.

```
import 'package:flutter/material.dart';
void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        body: Center(child: Text('Hello Flutter!', style: TextStyle(fontSize: 24))),
      ),
    );
  }
}
```

B) Display the fetched data in a meaningful way in the UI.

Display dynamic data from a variable or API.

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

void main() => runApp(FetchDisplayApp());

class FetchDisplayApp extends StatelessWidget {
  final String data = "Fetched Data: Flutter is amazing!";
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(body: Center(child: Text(data, style: TextStyle(fontSize: 20, color: Colors.blue))))
    );
  }
}
```

Question 3. A) Create Container in User Interface (UI) in Flutter.

Container used for layout, decoration, padding, and alignment.

```
import 'package:flutter/material.dart';
void main() => runApp(MyContainerApp());

class MyContainerApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        body: Center(
          child: Container(
            padding: EdgeInsets.all(20),
            color: Colors.teal,
            child: Text("Inside Container", style: TextStyle(color: Colors.white)),
          ),
        ),
      ),
    );
  }
}
```

B) Implement form validation and error handling.

Use Form and TextFormField with validator property.

```
import 'package:flutter/material.dart';
void main() => runApp(FormApp());

class FormApp extends StatelessWidget {
  final _formKey = GlobalKey<FormState>();

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        body: Padding(
          padding: EdgeInsets.all(20),
          child: Form(
            key: _formKey,
            child: Column(
              children: [
                TextFormField(validator: (v) => v == null || v.isEmpty ? 'Enter name' : null),
                SizedBox(height: 12),
                ElevatedButton(
                  onPressed: () {
                    if (_formKey.currentState!.validate()) {
                      debugPrint('Form valid');
                    }
                  },
                  child: Text('Submit'),
                )
              ],
            ),
          ),
        ),
```

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

Question 4. A) Display the data in Column by using column widget in Flutter.

Column arranges widgets vertically.

```
import 'package:flutter/material.dart';
void main() => runApp(ColumnApp());

class ColumnApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        body: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            Text('Name:'), TextField(),
            SizedBox(height:10),
            Text('Email:'), TextField(),
          ],
        ),
      ),
    );
  }
}
```

B) Create custom widgets for specific UI elements.

Define a reusable widget class extending StatelessWidget.

```
import 'package:flutter/material.dart';
void main() => runApp(CustomApp());

class MyButton extends StatelessWidget {
  final String label;
  const MyButton(this.label, {Key? key}) : super(key: key);
  @override
  Widget build(BuildContext context) {
    return ElevatedButton(onPressed: () {}, child: Text(label));
  }
}

class CustomApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(home: Scaffold(body: Center(child: MyButton("Click Me"))));
  }
}
```

Question 5. A) Implement layout structure using Row widget.

Row arranges widgets horizontally.

```
import 'package:flutter/material.dart';
void main() => runApp(RowApp());

class RowApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        body: Center(
          child: Row(
            mainAxisAlignment: MainAxisAlignment.spaceEvenly,
            children: [
              Icon(Icons.home), Icon(Icons.star), Icon(Icons.person)
            ],
          ),
        ),
      ),
    );
  }
}
```

B) Design a form with various input fields.

Use TextFields for multiple form inputs.

```
import 'package:flutter/material.dart';
void main() => runApp(FieldsApp());

class FieldsApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        body: Padding(
          padding: EdgeInsets.all(16),
          child: Column(children: [
            TextField(decoration: InputDecoration(labelText: 'Username')),
            TextField(obscureText: true, decoration: InputDecoration(labelText: 'Password')),
          ]),
        ),
      ),
    );
  }
}
```

Question 6. A) Create Stack widget layout structure in Flutter.

Stack overlays widgets on top of each other.

```
import 'package:flutter/material.dart';
void main() => runApp(StackApp());

class StackApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        body: Center(
          child: Stack(
            children: [
              Container(width: 200, height: 200, color: Colors.blue),
              Positioned(top: 50, left: 50, child: Icon(Icons.star, color: Colors.white, size: 50)),
            ],
          ),
        ),
      ),
    );
  }
}
```

B) Implement navigation with named routes.

Define routes inside MaterialApp and navigate using Navigator.pushNamed.

```
import 'package:flutter/material.dart';
void main() => runApp(RouteApp());

class RouteApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      initialRoute: '/',
      routes: {
        '/': (_) => Home(),
        '/next': (_) => NextPage(),
      },
    );
  }
}

class Home extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Center(
        child: ElevatedButton(
          child: Text('Go Next'),
          onPressed: () => Navigator.pushNamed(context, '/next'),
        ),
      ),
    );
  }
}
```

```
class NextPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) => Scaffold(body: Center(child: Text('Second Page')));
}
```

Question 7. A) Add animations to UI elements using Flutter's animation framework.

Use AnimatedOpacity, AnimatedContainer or AnimationController.

```
import 'package:flutter/material.dart';
void main() => runApp(AnimApp());

class AnimApp extends StatefulWidget {
  @override
  _AnimAppState createState() => _AnimAppState();
}

class _AnimAppState extends State<AnimApp> {
  double opacity = 0.0;
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        body: Center(
          child: AnimatedOpacity(
            opacity: opacity,
            duration: Duration(seconds: 2),
            child: Text('Animated Text'),
          ),
        ),
        floatingActionButton: FloatingActionButton(
          onPressed: () => setState(() => opacity = opacity == 0 ? 1 : 0),
          child: Icon(Icons.play_arrow),
        ),
      ),
    );
  }
}
```

B) Write unit tests for UI components.

Use flutter_test package and expect() for assertions.

```
// Save this as test/widget_test.dart and run `flutter test`
import 'package:flutter_test/flutter_test.dart';
void main() {
  test('Simple addition', () {
    expect(2 + 2, 4);
  });
}
```

Question 8. A) Implement fade text in Flutter.

Fade text using AnimatedOpacity widget.

```
import 'package:flutter/material.dart';
void main() => runApp(FadeApp());

class FadeApp extends StatefulWidget {
  @override
  _FadeAppState createState() => _FadeAppState();
}

class _FadeAppState extends State<FadeApp> {
  double _opacity = 1.0;
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        body: Center(
          child: AnimatedOpacity(opacity: _opacity, duration: Duration(seconds: 2), child: Text('Fade Ex
        ),
        floatingActionButton: FloatingActionButton(
          onPressed: () => setState(() => _opacity = _opacity == 0 ? 1 : 0),
          child: Icon(Icons.flip),
        ),
      ),
    );
  }
}
```

B) Set up navigation between different screens using Navigator.

Use Navigator.push() to move between screens.

```
import 'package:flutter/material.dart';
void main() => runApp(NavApp());

class NavApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(home: HomePage());
  }
}

class HomePage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Center(
        child: ElevatedButton(
          child: Text('Open Second'),
          onPressed: () => Navigator.push(context, MaterialPageRoute(builder: (_) => SecondPage())),
        ),
      ),
    );
  }
}
class SecondPage extends StatelessWidget {
```

```
  @override
  Widget build(BuildContext context) => Scaffold(body: Center(child: Text('Second Page')));
}
```

Question 9. A) Create text widget and apply slide animation.

Use SlideTransition with AnimationController.

```
import 'package:flutter/material.dart';
void main() => runApp(SlideApp());

class SlideApp extends StatefulWidget {
  @override
  _SlideAppState createState() => _SlideAppState();
}

class _SlideAppState extends State<SlideApp> with SingleTickerProviderStateMixin {
  late AnimationController ctrl;
  late Animation<Offset> slide;

  @override
  void initState() {
    super.initState();
    ctrl = AnimationController(vsync: this, duration: Duration(seconds: 1));
    slide = Tween(begin: Offset(0, 1), end: Offset.zero).animate(ctrl);
    ctrl.forward();
  }

  @override
  void dispose() {
    ctrl.dispose();
    super.dispose();
  }

  Widget build(BuildContext context) => MaterialApp(
        home: Scaffold(
          body: Center(
            child: SlideTransition(position: slide, child: Text('Sliding Text')),
          ),
        ),
      );
}
```

B) Implement media queries and breakpoints for responsiveness.

Use MediaQuery.of(context).size to get screen size.

```
import 'package:flutter/material.dart';
void main() => runApp(MediaQueryApp());

class MediaQueryApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        body: Builder(builder: (context) {
          double width = MediaQuery.of(context).size.width;
          return Center(child: Text(width < 600 ? 'Mobile View' : 'Tablet/Desktop View'));
        }),
      ),
    );
  }
```

```
}
```

Question 10. A) Implement state management using setState method.

setState rebuilds widget tree with updated data.

```
import 'package:flutter/material.dart';
void main() => runApp(StateApp());

class StateApp extends StatefulWidget {
  @override
  _StateAppState createState() => _StateAppState();
}

class _StateAppState extends State<StateApp> {
  int count = 0;
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        body: Center(child: Text('Count: $count', style: TextStyle(fontSize: 30))),
        floatingActionButton: FloatingActionButton(
          onPressed: () => setState(() => count++),
          child: Icon(Icons.add),
        ),
      ),
    );
  }
}
```

B) Design a responsive UI that adapts to different screen sizes.

Use LayoutBuilder to switch layout based on width.

```
import 'package:flutter/material.dart';
void main() => runApp(ResponsiveApp());

class ResponsiveApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        body: LayoutBuilder(builder: (context, constraints) {
          return Center(child: Text(constraints.maxWidth < 600 ? 'Mobile View' : 'Desktop View'));
        }),
      ),
    );
  }
}
```