योगः कर्मसु कौशलम्
IN PURSUIT OF PERFECTION

VIVEKANANDA INSTITUTE OF PROFESSIONAL STUDIES - TECHNICAL CAMPUS
Grade A++ Accredited Institution by NAAC
NBA Accredited for MCA Programme; Recognized under Section 2(f) by UGC;
Affiliated to GGSIP University, Delhi; Recognized by Bar Council of India and AICTE
An ISO 9001:2015 Certified Institution

SCHOOL OF ENGINEERING & TECHNOLOGY

BTECH Programme: IIOT

Course Title: Mobile Application Development Lab

Course Code: OAE312P

Submitted To: Dr. Alisha Sikri
Submitted By: Ayush Mangla
Enrollment no.: 03917711722

VIPS
योगः कर्मसु कौशलम्
IN PURSUIT OF PERFECTION

VIVEKANANDA INSTITUTE OF PROFESSIONAL STUDIES - TECHNICAL CAMPUS
Grade A++ Accredited Institution by NAAC
NBA Accredited for MCA Programme; Recognized under Section 2(f) by UGC;
Affiliated to GGSIP University, Delhi; Recognized by Bar Council of India and AICTE
An ISO 9001:2015 Certified Institution
SCHOOL OF ENGINEERING & TECHNOLOGY

VISION OF INSTITUTE

To be an educational institute that empowers the field of engineering to build a sustainable future by providing quality education with innovative practices that supports people, planet and profit.

MISSION OF INSTITUTE

To groom the future engineers by providing value-based education and awakening students' curiosity, nurturing creativity and building capabilities to enable them to make significant contributions to the world.

VIPS
योगः कर्मसु कौशलम्
IN PURSUIT OF PERFECTION

VIVEKANANDA INSTITUTE OF PROFESSIONAL STUDIES - TECHNICAL CAMPUS
Grade A++ Accredited Institution by NAAC
NBA Accredited for MCA Programme; Recognized under Section 2(f) by UGC;
Affiliated to GGSIP University, Delhi; Recognized by Bar Council of India and AICTE
An ISO 9001:2015 Certified Institution
SCHOOL OF ENGINEERING & TECHNOLOGY

INDEX

S.No	Experiment Name	Date	Marks			Remark	Updated Marks	Faculty Signature
			Laboratory Assessment (15 Marks)	Class Participation (5 Marks)	Viva (5 Marks)			

VIPS

योगः कर्मसु कौशलम्

IN PURSUIT OF PERFECTION

VIVEKANANDA INSTITUTE OF PROFESSIONAL STUDIES - TECHNICAL CAMPUS

Grade A++ Accredited Institution by NAAC

NBA Accredited for MCA Programme; Recognized under Section 2(f) by UGC;

Affiliated to GGSIP University, Delhi; Recognized by Bar Council of India and AICTE

An ISO 9001:2015 Certified Institution

SCHOOL OF ENGINEERING & TECHNOLOGY

Experiment 1

Aim: Hello World Application: Create a basic "Hello World" application for a mobile platform of your choice (Android or iOS) using the respective development environment.

Objectives:

- To Understand the basic development of Mobile Applications.
- To gain knowledge and practice of printing basic texts on a screen of mobile app.

Theory:

Introduction to Mobile Application Development
Mobile application development is the process of creating software applications that run on mobile devices such as smartphones and tablets. These applications can be developed for different platforms, primarily Android and iOS, using platform-specific or cross-platform development environments. A "Hello World" application is a fundamental starting point in mobile app development, serving as an introductory exercise to understand the development workflow, tools, and programming languages involved.

Development Environments for Mobile Platforms
Different platforms require different development environments:

- **Android Development:** Uses Android Studio with the Kotlin or Java programming language.
- **iOS Development:** Uses Xcode with the Swift programming language.
- **Cross-Platform Development:** Tools like Flutter (Dart) or React Native (JavaScript/TypeScript) allow development for both Android and iOS using a single codebase.

Fundamental Concepts in a Mobile Application
A basic "Hello World" application introduces key concepts in mobile development:

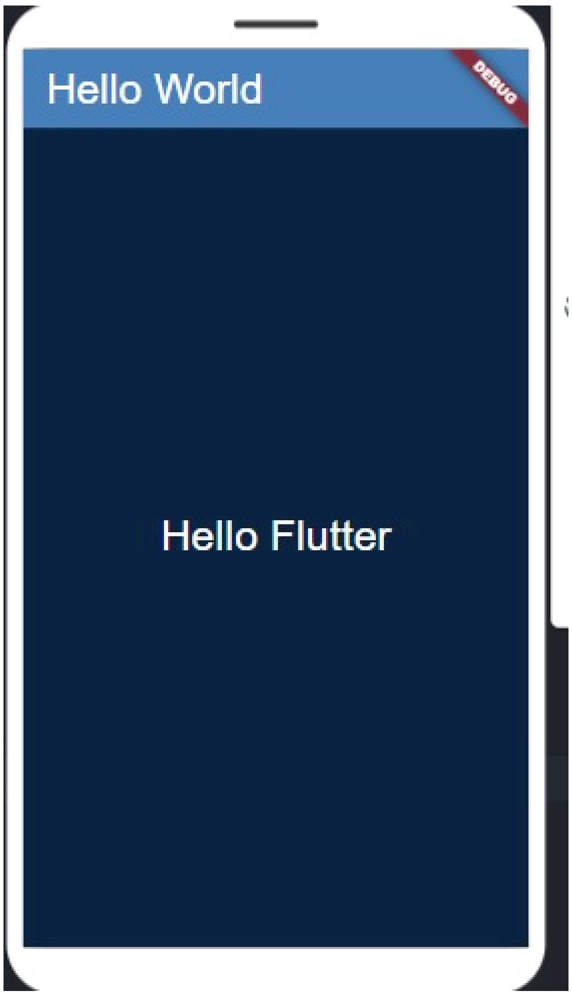
- **User Interface (UI):** The app's visual representation, usually defined using XML (Android), SwiftUI (iOS), or Flutter widgets.
- **Activity/ViewController:** The main screen or component that controls UI interaction.
- **Rendering & Lifecycle:** Understanding how the app initializes and displays content on the screen.

Creating a "Hello World" application is the first step in mobile development, offering insight into essential tools, frameworks, and programming structures. Whether using Android Studio, Xcode, or cross-platform frameworks like Flutter, this experiment builds confidence in developing mobile applications and sets the foundation for advanced mobile programming concepts.

Code:

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

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

class MyApp extends StatelessWidget {
  const MyApp({super.key});
  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      // Application name
      title: 'Flutter Hello World',
      // Application theme data, you can set the colors for the application as
      // you want
      theme: ThemeData(
        // useMaterial3: false,
        primarySwatch: Colors.blue,
      ),
      // A widget which will be started on application startup
      home: MyHomePage(title: 'FlutLab'),
    );
  }
}

class MyHomePage extends StatelessWidget {
  final String title;
  const MyHomePage({super.key, required this.title});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        // The title text which will be shown on the action bar
        title: Text(title),
      ),
      body: Center(
        child: Text(
          'Hello, World!',
        ),
      ),
    );
  }
}
```

Output:



Learning Outcomes:

Experiment 2

Aim: Design a simple mobile user interface to add image on a mobile platform.

Objectives:

- To understand the basic development of Mobile Applications.
- To gain knowledge and practice of displaying Images on a screen of mobile app.

Theory:

Introduction

Flutter is an open-source UI software development framework by Google used for building natively compiled applications for mobile, web, and desktop from a single codebase. One of the fundamental tasks in Flutter application development is displaying images, which can be achieved using widgets like `Image.asset()` and `Image.network()`.

Objective

The objective of this experiment is to understand the process of displaying an image in a Flutter application using both local assets and network sources. This experiment aims to familiarize developers with Flutter's image handling mechanisms, asset management, and UI design considerations.

Concept and Implementation

Flutter provides the `Image` widget as a simple way to render images on the screen. There are two primary ways to load images:

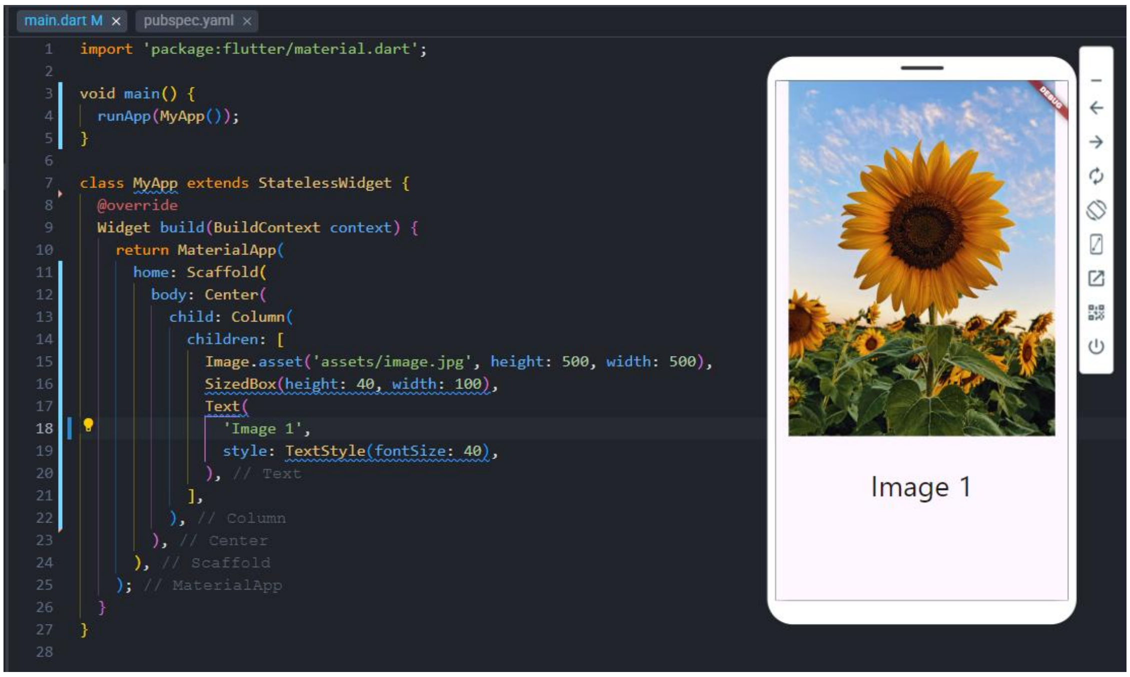
1. **Loading from Assets**: Images stored in the project's asset directory are displayed using the `Image.asset()` constructor. The asset must be declared in the `pubspec.yaml` file before use.
2. **Loading from a Network**: External images can be fetched from the internet using `Image.network()` by providing a valid URL.

This experiment demonstrates the ease of displaying images in a Flutter application using the Image widget. Understanding how to manage assets and load images dynamically enhances the ability to create visually rich and interactive applications. Mastering these fundamental concepts is essential for building advanced Flutter applications with dynamic content.

Code:

```dart
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: Column(
            children: [
              Image.asset('assets/image.jpg', height: 500, width: 500),
              SizedBox(height: 40, width: 100),
              Text(
                'Image 1',
                style: TextStyle(fontSize: 40),
              ),
            ],
          ),
        ),
      ),
    );
  }
}
```

Output:



main.dart M × pubspec.yaml ×

```dart
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: Column(
            children: [
              Image.asset('assets/image.jpg', height: 500, width: 500),
              SizedBox(height: 40, width: 100),
              Text(
                'Image 1',
                style: TextStyle(fontSize: 40),
              ), // Text
            ],
          ), // Column
        ), // Center
      ), // Scaffold
    ); // MaterialApp
  }
}
```

Learning Outcomes:

Experiment 3

Aim: Develop a mobile application that interacts with a button.

Objectives:

- To understand the basic development of Mobile Applications.
- To gain knowledge and practice of displaying buttons on a screen of mobile app for interaction.

Theory:

Introduction

Buttons are essential UI components in mobile applications that enable user interaction. In Flutter, buttons are widely used for triggering events, submitting forms, or navigating between screens. Buttons enhance the user experience by providing intuitive access to different functionalities. Flutter provides multiple types of buttons such as `ElevatedButton`, `TextButton`, `OutlinedButton`, and `IconButton`, each serving different UI requirements. Additionally, these buttons can be customized with different styles, colors, and animations to enhance usability and aesthetic appeal.

Objective

The objective of this experiment is to understand the implementation and functionality of buttons in a Flutter application. This involves creating various buttons, customizing their appearance, and handling user interactions through callback functions. Furthermore, this experiment explores how buttons contribute to UI/UX design by improving accessibility and user engagement.

Types of Buttons in Flutter

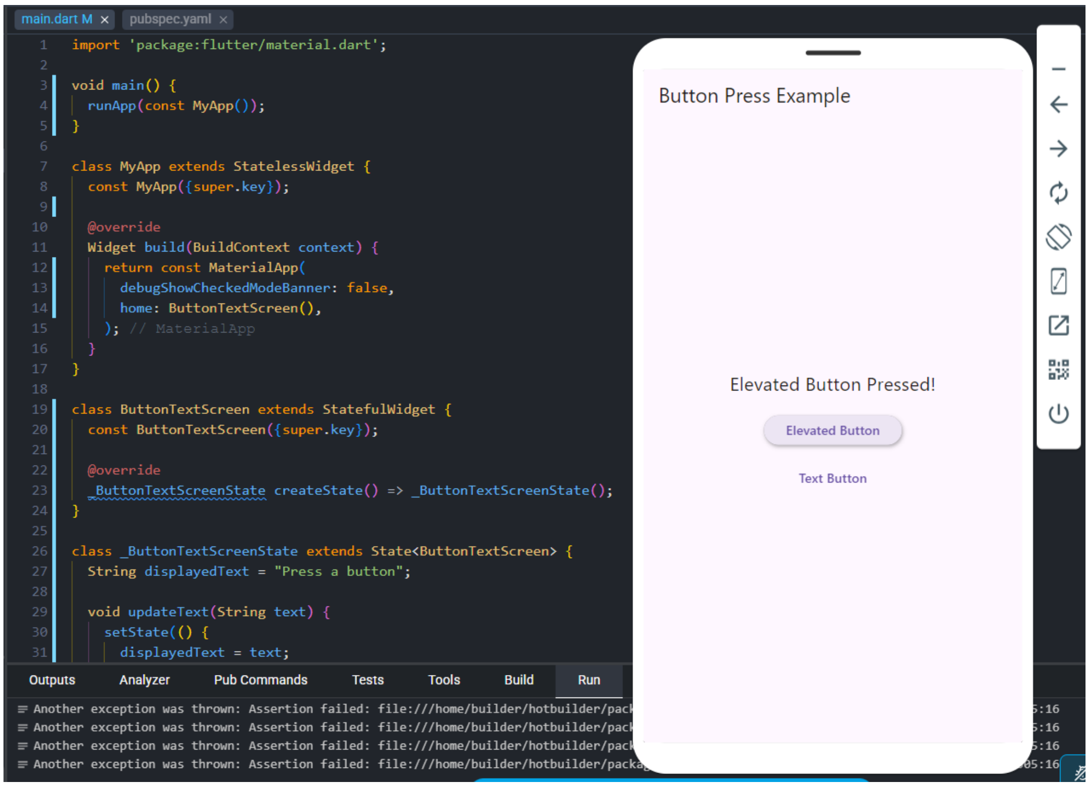
1. **ElevatedButton**: A button with a shadow effect, useful for emphasizing primary actions and adding depth to the UI.
2. **TextButton**: A simple button with no elevation, typically used for secondary actions or inline links.
3. **OutlinedButton**: A button with an outlined border, offering a balance between prominence and subtlety, often used when a softer emphasis is needed.
4. **IconButton**: A button displaying an icon instead of text, often used for toolbars, navigation, or quick actions.

Buttons play a crucial role in user interaction within a Flutter application. By understanding their types, properties, and usage, developers can create intuitive and user-friendly interfaces. Effective button placement and styling enhance the overall user experience, making applications more engaging and accessible. Mastering button implementation enables the development of dynamic and interactive applications, ensuring seamless interaction between the user and the app.

Code:

```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 const MaterialApp(
      debugShowCheckedModeBanner: false,
      home: ButtonTextScreen(),
    );
  }
}
class ButtonTextScreen extends StatefulWidget {
  const ButtonTextScreen({super.key});

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

class _ButtonTextScreenState extends State<ButtonTextScreen> {
  String displayedText = "Press a button";

  void updateText(String text) {
    setState(() {
      displayedText = text;
    });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('Button Press Example')),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            Text(
              displayedText,
              style: const TextStyle(fontSize: 20),
            ),
            const SizedBox(height: 20),
            ElevatedButton(
              onPressed: () => updateText("Elevated Button Pressed!"),
              child: const Text('Elevated Button'),
            ),
            const SizedBox(height: 20),
            TextButton(
              onPressed: () => updateText("Text Button Pressed!"),
              child: const Text('Text Button'),
            ),
          ],
        ),
      ),
    );
  }
}
```

Output:



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

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

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

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

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

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

class _ButtonTextScreenState extends State<ButtonTextScreen> {
  String displayedText = "Press a button";

  void updateText(String text) {
    setState(() {
      displayedText = text;
```

```
Another exception was thrown: Assertion failed: file:///home/builder/hotbuilder/pack
Another exception was thrown: Assertion failed: file:///home/builder/hotbuilder/pack
Another exception was thrown: Assertion failed: file:///home/builder/hotbuilder/pack
Another exception was thrown: Assertion failed: file:///home/builder/hotbuilder/packa
```

Learning Outcomes: