



Viva Questions for Exp 3

Sr. No	Viva Question	Answer
1	What is the purpose of Form widget?	To group multiple form fields and manage validation and saving.
2	How do you validate a form field in Flutter?	By using <code>validator</code> property of <code>TextFormField</code> .
3	What is <code>GlobalKey<FormState></code> ?	It is a key to uniquely identify and validate the Form.
4	How to submit a Form in Flutter?	Call <code>_formKey.currentState!.validate()</code> and <code>_formKey.currentState!.save()</code> .
5	What widget is used to take input from the user?	<code>TextFormField</code> widget.

Perfect! 

Your **Experiment 4** is:

"To design a layout of Flutter App using layout widgets."

That means:

→ You have to **create a nice layout** using Flutter widgets like `Column`, `Row`, `Container`, `Padding`, `Center`, `Image`, `Text`, `Icon`, etc.



Let's plan Perform task first:

I'll give you a simple + beautiful code which uses multiple **layout widgets**.



Experiment 4 Code: Layout using Flutter Widgets

dart

[CopyEdit](#)

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

```
void main() {  
  runApp(Exp4App());  
}
```

```

class Exp4App extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Experiment 4 - Layout Widgets',
      theme: ThemeData(
        primarySwatch: Colors.teal,
      ),
      home: Scaffold(
        appBar: AppBar(
          title: Text('My Flutter Layout App'),
          centerTitle: true,
        ),
        body: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            // Centered Image
            Center(
              child: Image.asset(
                'assets/images/flutter_logo.png',
                width: 150,
                height: 150,
              ),
            ),

            SizedBox(height: 20),

            // Some Text
            Text(
              'Welcome to Flutter!',
              style: TextStyle(fontSize: 24, fontWeight:
FontWeight.bold),
            ),

            SizedBox(height: 10),

            // Row with Icons
            Row(
              mainAxisAlignment: MainAxisAlignment.center,
              children: [
                Icon(Icons.favorite, color: Colors.red, size:
30),

                SizedBox(width: 10),

```

```

        Icon(Icons.star, color: Colors.yellow[700],
size: 30),
        SizedBox(width: 10),
        Icon(Icons.thumb_up, color: Colors.blue,
size: 30),
    ],
),

SizedBox(height: 20),

// A Button
ElevatedButton(
  onPressed: () {
    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(content: Text('Button Pressed!')),
    );
  },
  child: Text('Click Me'),
),
],
),
),
);
}
}

```

Expected Output:

When you run it, you will see:

- Flutter Logo (Image) in center
- Welcome Text
- Row of three colorful Icons
- A Button ("Click Me") → shows SnackBar when clicked

Important:

In your `pubspec.yaml`, don't forget:

```

yaml
CopyEdit
flutter:
  assets:

```

– assets/images/flutter_logo.png

→ And put any image (like Flutter logo) inside your project folder: assets/images/.

Viva Questions for Exp 4 (with answers):

Sr.	Question	Answer
1	What is a layout in Flutter?	Layout means arranging UI elements (like text, images, ...).
2	Name two basic layout widgets.	Row and Column.
3	What is the use of Column widget?	To arrange children vertically (top to bottom).
4	What is the use of Row widget?	To arrange children horizontally (left to right).
5	What is the purpose of	It is used for decorating, padding, and positioning child
6	How to add space between widgets?	Using SizedBox or Padding.
7	How to center a widget in Flutter?	Using Center widget.
8	What is mainAxisAlignment?	It controls how children are placed along the main axis (horizontal in Row, vertical in Column).
9	How to show an image in Flutter?	Using Image.asset('path ') or
10	What happens when you click the button in this experiment?	A SnackBar appears showing a message ("Button Pressed!").

now lets do exp 5 in detail with viva questions and how to perform the exp

ChatGPT said:

Experiment 5: To Include Icons, Images, Charts in Flutter App

Objective:

In this experiment, you will learn how to include icons, images, and charts into your Flutter app to enhance the visual appeal and functionality. This will involve using Flutter's built-in widgets for images and icons, and integrating third-party packages for charts.

Steps to Perform the Experiment

Step 1: Setting up the Flutter Project

1. **Create a New Flutter Project:** Open your terminal and run the following commands to create a new Flutter project.

```
bash
CopyEdit
```

```
flutter create icons_images_charts
```

2. `cd icons_images_charts`
- 3.

4. **Open the Project:** Open the project folder in your preferred IDE (e.g., VS Code or Android Studio).

Step 2: Add Dependencies for Charts

To include charts, we need to use a Flutter package called `fl_chart` which provides various types of charts like line, bar, pie, etc.

1. **Open `pubspec.yaml`** in your project and add the following dependencies under `dependencies::`

`yaml`

[CopyEdit](#)

`dependencies:`

2. `flutter:`
3. `sdk: flutter`
4. `fl_chart: ^0.40.0 # For charts`
- 5.

6. **Run `flutter pub get`:** This will download and install the necessary packages.

Step 3: Adding Icons and Images

1. **Adding Icons:** Flutter comes with a built-in set of Material icons. You can use them directly by referring to their names.

Example:

`dart`

[CopyEdit](#)

`Icon(Icons.home, size: 40.0, color: Colors.blue)`

- 2.

You can use different types of icons from the `Icons` class, such as `Icons.search`, `Icons.favorite`, etc.

3. **Adding Images:** To use images in your app, you need to add the images to the `assets` folder.

Steps:

- Create a new folder named `assets/images` in your project.
- Place your image files (e.g., `image1.jpg`, `image2.png`) into this folder.

4. Now, register the images in the `pubspec.yaml` file:

```
yaml
CopyEdit
```

```
flutter:
```

5. `assets:`
6. `- assets/images/`
- 7.

Use the image in your code like this:

```
dart
CopyEdit
```

```
Image.asset('assets/images/image1.jpg', width: 100,
height: 100)
```

- 8.

Step 4: Adding Charts

For this, we'll use the `fl_chart` package to add a simple **Line Chart** to the Flutter app.

Example for Line Chart:

1. **Import the necessary package:**

```
dart
CopyEdit
```

```
import 'package:fl_chart/fl_chart.dart';
```

- 2.

3. Create a basic Line Chart:

Replace the code in `lib/main.dart` with the following:

dart

CopyEdit

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

4. import 'package:fl_chart/fl_chart.dart';
5.
6. void main() {
7.   runApp(MyApp());
8. }
9.
10. class MyApp extends StatelessWidget {
11.   @override
12.   Widget build(BuildContext context) {
13.     return MaterialApp(
14.       title: 'Flutter Icons, Images, and Charts',
15.       theme: ThemeData(
16.         primarySwatch: Colors.blue,
17.       ),
18.       home: HomePage(),
19.     );
20.   }
21. }
22.
23. class HomePage extends StatelessWidget {
24.   @override
25.   Widget build(BuildContext context) {
26.     return Scaffold(
27.       appBar: AppBar(
28.         title: Text('Icons, Images, and Charts'),
29.       ),
30.       body: SingleChildScrollView(
31.         child: Column(
32.           children: <Widget>[
33.             // Using an icon
34.             Icon(Icons.home, size: 60, color:
35. Colors.blue),
36.             // Using an image
37.             Image.asset('assets/images/image1.jpg',
38. width: 300, height: 300),
```

```

39.         // Adding Line Chart
40.     Padding(
41.         padding: const EdgeInsets.all(16.0),
42.         child: LineChart(
43.             LineChartData(
44.                 gridData: FlGridData(show: true),
45.                 titlesData: FlTitlesData(show: true),
46.                 borderData: FlBorderData(show: true),
47.                 lineBarsData: [
48.                     LineChartBarData(
49.                         spots: [
50.                             FlSpot(0, 1),
51.                             FlSpot(1, 3),
52.                             FlSpot(2, 1.5),
53.                             FlSpot(3, 2.5),
54.                             FlSpot(4, 4),
55.                         ],
56.                         isCurved: true,
57.                         colors: [Colors.blue],
58.                         barWidth: 4,
59.                         belowBarData: BarAreaData(show:
60. false),
61.                     ),
62.                 ],
63.             ),
64.         ),
65.     ],
66. ),
67. );
68. );
69. }
70. }
71.

```

Explanation:

- **Icon:** This is the home icon, displayed with size 60 and blue color.
- **Image:** An image is displayed from the assets folder (replace `image1.jpg` with your own image).
- **Chart:** A simple line chart is created using `fl_chart`. The data points are represented as `FlSpot` objects, and the chart is styled using properties like `isCurved`, `colors`, and `barWidth`.

Step 5: Run the Application

After you've set up everything:

1. Run the app using the following command:

```
bash
```

```
CopyEdit
```

```
flutter run
```

- 2.
3. You should see:
 - A **home icon** at the top.
 - An **image** below the icon.
 - A **line chart** showing some sample data.

Viva Questions for Experiment 5

1. **What is the purpose of using icons in a mobile application?**
 - Icons help to visually represent actions or features in the app, making it easier for users to navigate and interact with the app.
2. **How do you add external images to a Flutter project?**
 - External images are added by placing them in the `assets` folder and registering them in the `pubspec.yaml` file.
3. **What is the role of the `fl_chart` package in this experiment?**
 - The `fl_chart` package is used to create charts (line charts, bar charts, etc.) in a Flutter application. It helps to visualize data in a graphical format.
4. **Explain the `F1Spot` class in the `fl_chart` package.**
 - The `F1Spot` class represents a single data point in a chart. It takes two parameters: the x and y coordinates of the data point.
5. **How do you customize the appearance of a chart in Flutter?**
 - You can customize the chart by changing properties like the chart's border, grid lines, title, bar width, color, etc., in the `LineChartData` and `LineChartBarData` classes.

6. **How do you handle user input for dynamic chart data in Flutter?**
 - You can use a stateful widget and update the chart data by modifying the list of `FlSpot` objects based on user input or real-time data.
7. **What are the different types of charts you can create with the `fl_chart` package?**
 - Line charts, bar charts, pie charts, radar charts, and scatter plots are some of the available chart types.
8. **What is the significance of the `barWidth` property in the chart?**
 - The `barWidth` property determines the thickness of the lines or bars in the chart.

Viva Questions for Experiment 6

Here are some potential questions that you may encounter during your viva:

Basic Questions:

1. **What is the Navigator widget in Flutter?**
 - The `Navigator` widget in Flutter manages a stack of routes and allows you to move between them. It provides methods like `push`, `pop`, and `pushNamed` to navigate between pages.
2. **Explain how `Navigator.pushNamed()` works in Flutter.**
 - `Navigator.pushNamed()` is used to push a new route onto the navigation stack. It takes a route name as a string, which corresponds to the route defined in the `MaterialApp`'s `routes` property.
3. **What is the role of `GestureDetector` in Flutter?**
 - `GestureDetector` is a widget that detects gestures like tap, swipe, drag, and pinch. You can use it to add interactivity to widgets in Flutter.

4. What are the different gestures that can be handled by **GestureDetector**?

- **GestureDetector** can handle several gestures such as:
 - **onTap**: For tap events.
 - **onLongPress**: For long press events.
 - **onHorizontalDragUpdate**: For horizontal drag events.
 - **onVerticalDragUpdate**: For vertical drag events.
 - **onPanUpdate**: For free movement drag events.

Intermediate Questions:

5. How does **Navigator.pop()** work in Flutter?

- **Navigator.pop()** removes the top route from the navigation stack, effectively going back to the previous screen.

6. What is the difference between **Navigator.push()** and **Navigator.pushNamed()**?

- **Navigator.push()** allows you to push a route by directly passing a **Widget**, whereas **Navigator.pushNamed()** uses a named route defined in the **MaterialApp** widget.

7. Explain the use of the **initialRoute** property in the **MaterialApp** widget.

- The **initialRoute** property specifies the first route (screen) that will be displayed when the app is launched. It is used to set a default screen.

Advanced Questions:

8. What are some other navigation techniques in Flutter, apart from **Navigator.push()**?

- Apart from **Navigator.push()** and **Navigator.pushNamed()**, Flutter provides other navigation methods like:
 - **Navigator.pushReplacement()**: Replaces the current route with a new one.
 - **Navigator.pushAndRemoveUntil()**: Pushes a route and removes all the previous routes until a condition is met.

9. How would you handle deep linking in Flutter for specific routes?

- Deep linking in Flutter can be handled by using `onGenerateRoute` in the `MaterialApp` widget, which allows you to define custom route logic for deep links

To implement the above code for navigation, routing, and gestures in Flutter, you will add the code into the appropriate files in your Flutter project. Here's a step-by-step guide on where to add the code:

1. Create a New Flutter Project

If you haven't already created a Flutter project, open your terminal and run:

```
bash
```

```
CopyEdit
```

```
flutter create gesture_navigation_app
```

```
cd gesture_navigation_app
```

After creating the project, open the project in your preferred IDE (e.g., VS Code or Android Studio).

2. Add Code to `lib/main.dart`

The `main.dart` file is where you'll add all the code for your app. This file is automatically created when you create a Flutter project. Here's how to proceed:

File Structure

1. **`lib/main.dart`:** This file is where the core logic of your app, including navigation and gestures, will reside.

3. Full Code for `lib/main.dart`

Code for Routing and Navigation:

Open the `lib/main.dart` file and replace the existing code with the following code. This will set up the routes, navigation, and the `GestureDetector` for detecting taps and swipes.

```
dart
```

```
CopyEdit
```

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

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

```
class MyApp extends StatelessWidget {  
  @override
```

```

Widget build(BuildContext context) {
  return MaterialApp(
    title: 'Flutter Navigation & Gestures',
    initialRoute: '/',
    routes: {
      '/': (context) => HomePage(),
      '/second': (context) => SecondPage(),
    },
  );
}

class HomePage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Home Page'),
      ),
      body: Center(
        child: GestureDetector(
          onTap: () {
            print('Tapped!');
            Navigator.pushNamed(context, '/second');
          },
          onHorizontalDragUpdate: (details) {
            print('Swiped horizontally: $
{details.localPosition}');
          },
          child: Container(
            padding: EdgeInsets.all(20),
            color: Colors.blue,
            child: Text(
              'Tap to go to second page',
              style: TextStyle(color: Colors.white, fontSize:
20),
            ),
          ),
        ),
      ),
    );
}

class SecondPage extends StatelessWidget {

```

```

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text('Second Page'),
    ),
    body: Center(
      child: ElevatedButton(
        onPressed: () {
          // Navigate back to the home page
          Navigator.pop(context);
        },
        child: Text('Back to Home Page'),
      ),
    ),
  );
}
}

```

Explanation:

- **MyApp class:** This is the root widget where you set up the routes for the app. The **MaterialApp** widget takes care of the navigation and routing. It has an **initialRoute** of `'/'` (the home page) and a map of named routes (`'/'` and `'/second'`).
- **HomePage class:** This is the home page of the app. It contains a **GestureDetector** widget that listens for tap and horizontal drag gestures. When you tap on the screen, it navigates to the **SecondPage**. When you swipe horizontally, it prints the drag details.
- **SecondPage class:** This is the second page of the app, which has a button that allows you to go back to the home page by calling `Navigator.pop(context)`.

4. Run the App

After adding the code in the `lib/main.dart` file, it's time to run the app. In your terminal (inside the project directory), run the following command:

```
bash
```

```
CopyEdit
```

```
flutter run
```

This will build and run the app on your emulator or physical device.

You should now be able to:

- **Tap** on the home page to navigate to the second page.

- **Swipe** horizontally to see the drag event print in the console.

5. Testing Navigation and Gestures

1. **Navigate to the second page:** Tap the blue container on the home page, which will navigate to the second page.
2. **Swipe gesture:** On the home page, swipe horizontally and observe the drag update printed in the console.
3. **Navigate back:** On the second page, press the "Back to Home Page" button to navigate back to the home page.

Conclusion

By adding the above code to the `lib/main.dart` file, you have set up the navigation and gesture detection functionality in your Flutter app. The navigation allows you to move between pages, and the gestures (like tap and swipe) add interactivity.

Let me know if you face any issues or need more explanation!



Viva Questions for Experiment 9

Sr No.	Viva Question	Answer
1	What is Material UI?	Material UI is a popular React UI framework based on Google's Material Design principles.
2	What is the purpose of Grid in Material UI?	Grid is used to create responsive layouts that adjust based on the screen size.
3	How to make an E-Commerce UI responsive?	By using Grid, Box, Container components from Material UI which support responsive breakpoints (xs, sm, md, lg).
4	Why React with Material UI for E-Commerce Apps?	React provides component-based structure; Material UI offers ready-to-use design components — speeding up development.
5	How can you test responsiveness?	Use browser Developer Tools (Ctrl+Shift+I) and check the UI for mobile, tablet, and desktop screens.
6	What is the use of AppBar and Toolbar in Material UI?	AppBar creates a top bar (navbar); Toolbar arranges content inside the AppBar.



Bonus: Viva Questions for Experiment 10

Question	Answer
What is a Web App Manifest?	It's a JSON file that provides metadata about a PWA to browsers, enabling "Add to Home Screen" feature.
What are important fields in the manifest?	name, short_name, start_url, display, theme_color, background_color, icons.
Why are icons important in manifest?	Icons are used when the app is added to a mobile device's home screen.
What is display: standalone ?	It makes the app look like a native app without browser UI (address bar, etc.).
Where do you link manifest file in your app?	Inside the <code><head></code> tag of <code>index.html</code> using <code><link rel="manifest" href="manifest.json"></code> .
Do you need HTTPS for PWA features?	Yes, PWAs must be served over HTTPS (except on localhost for development).



Experiment 1

Title:

To install and configure Flutter Environment



How to Perform:



Steps to Install Flutter:

1. Download Flutter SDK:

- Go to flutter.dev
- Download **Flutter stable version** for your OS (Windows/Mac/Linux).

2. Extract the zip file:

- Extract Flutter SDK to a folder like `C:\src\flutter` (Windows) or `/Users/yourname/flutter` (Mac).

3. Set Environment Variable:

- Add Flutter's `bin/` directory to your System PATH. Example: `C:\src\flutter\bin`

4. Install Android Studio:

- Install **Android Studio**.
- Add **Flutter** and **Dart** plugins from Settings > Plugins.

5. Install Chrome (for web) and VS Code (optional):

- Install Chrome for web app testing.
- Install VS Code + Flutter extension (if you like lightweight editor).

6. Check Installation: Open Terminal or CMD and run:

```
bash
```

```
CopyEdit
```

```
flutter doctor
```

✅ It will show installed components (Flutter SDK, Android Studio, Device, etc.)

✅ Completion Requirements:

- `flutter doctor` must show ✅ all OK.
- Android Studio or VS Code installed.
- Device/Emulator/Chrome set up.
- Created first sample project.

Experiment 7

Title:

To Connect Flutter UI with Firebase Database

How to Perform:

✅ Steps to Connect Firebase:

1. Create Firebase Project:

- Go to console.firebase.google.com

- Create a new project → name it → continue.

2. Add Android App to Firebase:

- Register app (example `com.example.exp7firebase`).
- Download `google-services.json` file.

3. Setup in Flutter Project:

- Create Flutter project.
- Place `google-services.json` inside `android/app/`.
- In `android/build.gradle`:

gradle

CopyEdit

```
dependencies {  
    classpath 'com.google.gms:google-services:4.3.10' // add  
}
```

- In `android/app/build.gradle`:

gradle

CopyEdit

```
apply plugin: 'com.google.gms.google-services' // add at  
bottom
```

4. Add dependencies in `pubspec.yaml`:

yaml

CopyEdit

```
dependencies:  
  firebase_core: ^2.4.0  
  cloud_firestore: ^4.5.0
```

5. Initialize Firebase:

- In `main.dart`:

dart

CopyEdit

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

```
void main() async {  
  WidgetsFlutterBinding.ensureInitialized();  
  await Firebase.initializeApp();  
  runApp(MyApp());  
}
```

```
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(title: Text('Firebase Demo')),
        body: Center(child: Text('Connected to Firebase!')),
      ),
    );
  }
}
```

✅ Completion Requirements:

- Flutter project successfully connected to Firebase.
- `firebase_core` initialized.
- Firestore read/write operations working (optional small demo).

Experiment 8

Title:


To test and deploy production ready Flutter App on Android platform

How to Perform:

✅ Steps to Deploy Flutter App on Android:

1. Complete your Flutter app (UI + Logic ready).
2. Generate Signed APK:
 - Create a keystore (for real app):

bash

 **CopyEdit**

```
keytool -genkey -v -keystore my-release-key.jks -keyalg RSA
-keysize 2048 -validity 10000 -alias my-key-alias
```

(Fill password, name, etc.)

- Save `my-release-key.jks` in `android/app/`.
- Edit `android/key.properties`:

properties

[CopyEdit](#)

```
storePassword=your-password
keyPassword=your-password
keyAlias=my-key-alias
storeFile=../app/my-release-key.jks
• Edit android/app/build.gradle:
```

gradle

[CopyEdit](#)

```
android {
    ...
    defaultConfig { ... }
    signingConfigs {
        release {
            keyAlias keystoreProperties['keyAlias']
            keyPassword keystoreProperties['keyPassword']
            storeFile file(keystoreProperties['storeFile'])
            storePassword keystoreProperties['storePassword']
        }
    }
    buildTypes {
        release {
            signingConfig signingConfigs.release
            minifyEnabled false
            shrinkResources false
        }
    }
}
```

3. Build Release APK:

bash

[CopyEdit](#)

```
flutter build apk --release
```

✅ APK will be generated inside:

```
build/app/outputs/flutter-apk/app-release.apk
```

4. Test on Device:

- Install APK manually on your Android device.
- Test all features working correctly (buttons, navigation, firebase if used).

5. Deployment (Optional):

- Upload to Google Play Store using Developer Account.

✓ Completion Requirements:

- Generated APK.
- App installed and tested on a real Android phone.
- (Optional) App uploaded to Play Store if needed.

Viva Questions:

Sr No.	Question	Answer
1	What is Flutter?	Flutter is a UI toolkit for building natively compiled applications for mobile, web, and desktop from a single codebase.
2	What are the prerequisites to install Flutter?	Flutter SDK, Android Studio/VS Code, JDK, Chrome (for web apps).
3	What does <code>flutter doctor</code> command do?	It checks your environment and displays a report of the installation status.
4	What is the use of Dart in Flutter?	Dart is the programming language used to develop Flutter apps.
5	Name two popular editors for Flutter development.	

Sr No.	Question	Answer
1	What is Firebase?	Firebase is a platform developed by Google for creating mobile and web applications.
2	Why do we use <code>firebase_core</code> package?	It is necessary to initialize Firebase services in a Flutter app.
3	What file connects Flutter Android app to Firebase?	<code>google-services.json</code> file.
4	Name any two Firebase services.	Firebase Authentication, Firebase Cloud Firestore.
5	How do you initialize Firebase in Flutter?	Using <code>await Firebase.initializeApp()</code> inside <code>main()</code> method.

Viva Questions:

Sr No.	Question	Answer
1	What is an APK?	APK (Android Package Kit) is the file format used to distribute and install applications on Android.
2	Why do we need to generate a release APK?	Release APK is optimized and signed for production and store deployment.
3	Which command is used to build a release APK?	<code>flutter build apk --release</code>
4	What is a keystore file?	It is a file that contains security keys used to sign the APK.
5	How to test the APK on a real device?	Transfer the APK to the device and manually install it to test.