Flutter apps use **Dart** programming language for creating an app.

Flutter doctor command checks your environment and displays a report of the status of your Flutter installation.

Every Dart program requires the top-level **main()** function, where execution starts. 1

Row widget arranges its children in a horizontal direction on the screen.

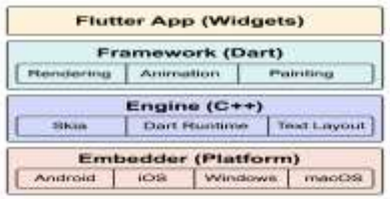
What is the use of pubspec.yaml file?
- The pubspec.yaml file is used to manage a Flutter project's metadata, dependencies, assets, and configuration. - It defines the project name, version, packages required, and resources like images or fonts used in the app. 1

Explain Hot Reload feature of flutter.
- The Hot Reload feature in Flutter allows developers to instantly see the changes made in the code without restarting the entire app. - It quickly updates the UI and preserves the current app state, making development and debugging faster and more efficient. 1

Explain loops in Dart with example.
- Loops in Dart are used to execute a block of code repeatedly until a specific condition is met. Dart provides different types of loops such as for loop, while loop, and do-while loop. **Example:** 1
void main() { for (int i = 1; i <= 5; i++) {print(i);}}
- In this example, the for loop prints numbers from 1 to 5 by increasing the value of i each time until the condition i <= 5 becomes false.

Explain Dart libraries.
Dart libraries are collections of reusable code that help organize programs and make them modular. They allow developers to use predefined functions, classes, and constants without rewriting code. Dart has built-in libraries like dart:core, dart:math, and dart:io, and developers can also create custom libraries or import external packages using the import statement. **Example:** 1
import 'dart:math';
void main() {print(sqrt(16)); // Output: 4.0} - Here, the dart:math library is imported to use the sqrt() function.

Explain Flutter Architecture with diagram.
Flutter architecture is based on a layered structure that helps in building high-performance, cross-platform applications.
It mainly consists of three layers:
1. Framework Layer: This layer is written in Dart and provides a rich set of widgets, animation, painting, and gestures. It includes components like Material and Cupertino libraries to create UI elements.
2. Engine Layer: The engine is written in C++ and provides low-level rendering support using Skia, text layout, and plugin handling. It acts as a bridge between the framework and the underlying platform. 1
3. Embedder Layer: The embedder connects Flutter with the specific platform (like Android, iOS, Windows, etc.). It handles input, output, and manages the app lifecycle and rendering surface.
In summary, Flutter's layered architecture ensures fast rendering, reactive UI updates, and smooth cross-platform performance. **Diagram:**



What are the Basic Flutter Widgets? Explain any two.
Flutter provides several basic widgets that are the building blocks of every Flutter app. These widgets are used to create the user interface and handle user interactions. Some of the basic Flutter widgets are: Text, Container, Row, Column, Image, Icon, Scaffold
Explanation of any two: 1
1.Text Widget: The Text widget is used to display a string of text with single style. You can customize the text's color, font size, weight, and alignment. Example: Text('Hello Flutter',style: TextStyle(fontSize: 20, color: Colors.blue),);
2.Container Widget : The Container widget is used to create a rectangular visual element that can contain other widgets. It allows setting padding, margin, color, border, and alignment.
Example:Container(padding: EdgeInsets.all(10),color: Colors.green,child: Text('This is a container'),);
These widgets form the foundation of Flutter's UI design and help in creating responsive layouts.

Explain Internationalisation and Localisation in Flutter:
Internationalisation (i18n) prepares an app for multiple languages, while Localisation (l10n) translates content based on the user's region. 5
1. Add flutter_localizations and intl packages.
2. Set supported locales in MaterialApp: supportedLocales: [Locale('en'), Locale('hi')]
3. Create .arb files for each language.
Example: en.arb: { "hello": "Hello" }
The app automatically displays text in the user's selected language.

Responsive app is one that is written in such a way that it fits the available screen size of the device it is viewed on. 2

AnimatedContainer widget is the animated version of Container widget that gradually changes its values over a period of time.

Navigator.push() method is used to navigate from one route to another route.

GridView is a widget in Flutter that displays the items in two-dimensional rows and columns. — **True**

What is MediaQuery?
MediaQuery is a Flutter widget that provides information about the size, orientation, and other properties of the device's screen. It helps in creating responsive layouts that adapt to different screen sizes and orientations. Example: var width = MediaQuery.of(context).size.width; 2

How to debug UI issues in Flutter?
UI issues in Flutter can be debugged using tools like: - Flutter Inspector to analyze the widget tree and layout. - Debug painting (debugPaintSizeEnabled = true) to visualize widget boundaries. - Hot Reload to quickly apply changes and test fixes.
- Console logs and breakpoints to trace problems in the code. 2

Difference between Stateless and Stateful Widgets:
Stateless Widget:
- A widget that does not change once built.
- No internal state is maintained.
- Rebuilt only when external data changes.
- Eg: Text, Icon, Container 2
Stateful Widget :
- A widget that can change its state during runtime.
- Maintains a state using the State class.
- Can rebuild itself when its internal state changes using setState().
- Eg: Checkbox, TextField, Slider

Explain SliverAppBar Widget:
The SliverAppBar is a Flutter widget that provides a flexible and scrollable app bar that can expand, collapse, or float as the user scrolls. It is typically used inside a CustomScrollView.
Example: SliverAppBar(title: Text('My App'), expandedHeight: 200.0, flexibleSpace: FlexibleSpaceBar(background: Image.asset('assets/header.jpg', fit: BoxFit.cover),), floating: true, pinned: true,) 2
This allows the app bar to expand with an image and remain visible or partially visible while scrolling.

Explain the Layout and Widgets Hierarchy with Example.
In Flutter, everything is a widget. Widgets form a tree-like structure called the widget hierarchy. Each widget is nested inside another to build the complete layout. **Example:**
MaterialApp(home: Scaffold(appBar: AppBar(title: Text('Demo')), body: Column(children: [Text('Hello'), Icon(Icons.star),],),),); 2
Explanation:
MaterialApp → Root widget
Scaffold → Page structure
Column → Layout manager
Text, Icon → Leaf widgets

Explain Custom Widget Creation in Flutter:
Custom widgets help reuse UI components and keep code clean. You can create one by extending StatelessWidget or StatefulWidget.
Example:
class MyText extends StatelessWidget {final String name; MyText(this.name); Widget build(BuildContext context) {return Text('Hello $name');}} 2
Explanation: This creates a simple reusable text widget that displays "Hello Flutter."

1.BloC stands for **Business Logic Component**.
2. Flutter is not declarative — **False**.
3. **App state** type of state can be used globally.
4. BaaS stands for **Backend as a Service.**

What are DevTools?
DevTools are a set of performance and 3
debugging tools provided by Flutter. They help developers inspect the widget tree, analyze app performance, debug layout issues, and track memory and network usage.

What is InheritedWidget?
InheritedWidget is a special type of widget in Flutter that allows data to be efficiently 3
passed down the widget tree without using constructors. It is commonly used for state management, where child widgets can access shared data from a common ancestor.

Explain Stream-based State Management with RxDart:
Stream-based state management uses Streams and RxDart to handle data flow reactively in Flutter apps. A stream provides asynchronous data that widgets can listen to and rebuild automatically when data changes. RxDart adds more features like BehaviorSubject to manage the latest value and broadcast updates.
Example: final counter = 3
BehaviorSubject<int>.seeded(0);
counter.add(counter.value + 1);
This helps keep UI and data in sync using reactive programming.

Write a Note on ScopedModel:
ScopedModel is a simple state management technique in Flutter that allows data sharing between widgets without manual passing. It holds the app state in a model class and notifies all listening widgets when data changes.
Example: class CounterModel extends Model {int count = 0; void increment() {count++; notifyListeners();}} 3
ScopedModel is easy to use for small to medium apps but is less preferred now compared to Provider or Riverpod.

Explain BLoC Pattern for State Management:
BLoC (Business Logic Component) is a state management pattern that separates business logic from the UI. It uses Streams and Sinks to handle data flow between the UI and logic layers.
This makes the app more scalable, testable, and easier to maintain. 3
Main Components:
Event – User actions (like button clicks).
State – The current data shown on the UI.
BLoC – Receives events, processes logic, and emits new states through streams.
Flow: UI → Event → BLoC → Stream → UI (updated state)
Example: class CounterBloc {final _counter = StreamController<int>(); int _count = 0; Stream<int> get counterStream => _counter.stream; void increment() {_count++; _counter.sink.add(_count);} void dispose() => _counter.close();}
Here, the UI listens to `counterStream` and updates automatically when `increment()` is called.

Explain Redux Architecture for Flutter:
Redux is a predictable state management architecture based on a single global store that holds the entire app state. It uses a unidirectional data flow.
Main Components: 3
1.Store – Holds the application state.
2.Action – Describes what happened (like increment, delete, update).
3.Reducer – A pure function that takes the current state and an action, and returns a new state.
4.View – Displays the state and dispatches actions.
Flow: View → Action → Reducer → Store → View (updated)
Example:
int counterReducer(int state, dynamic action) {if (action == 'INCREMENT') return state + 1; return state;}
Redux helps maintain a single source of truth, making the app easier to debug and scale — especially useful for large Flutter projects.

1. Flutter provides **http** package to use HTTP resources.
2. JSON stands for **JavaScript Object Notation.** 4
3. Flutter apps can make use of the SQLite databases via **sqflite** plugin available on pub.dev.
4. REST stands for **Representational State Transfer.**

What is JSON Serialization?
JSON serialization is the process of converting Dart objects into JSON format so they can be easily stored or sent over a network. Similarly, deserialization converts JSON data back into Dart objects. It is commonly used for API 4
communication in Flutter apps.
Example: var jsonData = jsonEncode({'name': 'John', 'age': 25});

What is Hive in Flutter?
Hive is a lightweight and fast local database for Flutter. It is a NoSQL database that stores data in key-value pairs and works efficiently without requiring a device-specific setup. It's useful for 4
offline storage and caching data locally.

Explain Caching Data in Flutter:
Caching in Flutter means storing data locally so the app can access it quickly without needing to fetch it repeatedly from the internet or a database. It improves performance and allows offline access.
-Data can be cached using tools like SharedPreferences, Hive, or sqflite.
Example: SharedPreferences prefs = await SharedPreferences.getInstance(); 4
prefs.setString('username', 'Madhav');
Here, data is saved locally and can be retrieved later even if the app restarts.

Explain the Use of WebSockets for Real-Time Communication in Flutter:
WebSockets enable two-way real-time communication between a Flutter app and a server. Unlike HTTP, which requires repeated requests, WebSockets maintain a continuous connection, making it ideal for chat apps, live updates, or notifications.
Example: var socket = 4
WebSocket.connect('ws://example.com');
The app can both send and receive messages instantly, ensuring live interaction without delay.

Explain Shared Preferences and Secure Storage in Flutter:
Shared Preferences and Secure Storage are used to store data locally on a device. - Shared Preferences is for small non-sensitive data like settings or login info. **Example**: var prefs = await SharedPreferences.getInstance(); prefs.setInt('age', 20);
-Secure Storage is used for sensitive data like passwords or tokens because it encrypts data. **Example**: var storage = FlutterSecureStorage(); await storage.write(key: 'token', 4
value: 'abc123'); - Shared Preferences is simple but not secure, while Secure Storage provides encryption for safety.

How to Perform CRUD Operations on a SQLite Database in Flutter:
SQLite is used to store structured data locally. CRUD means Create, Read, Update, and Delete.
Use the sqflite package to perform these operations.
Example: var db = await openDatabase('user.db', version: 1, onCreate: (db, v) => db.execute('CREATE TABLE user(id INTEGER, name TEXT)'));
await db.insert('user', {'id': 1, 'name': 'Raj'}); // Create 4
var data = await db.query('user'); // Read
await db.update('user', {'name': 'Amit'}, where: 'id=?', whereArgs: [1]); // Update
await db.delete('user', where: 'id=?', whereArgs: [1]); // Delete
- SQLite helps store and manage app data offline in a structured and reliable way.

1. **Tween animation** in Flutter is used to define how values change smoothly between a begin and an end value over time.
2. **location** plugin in Flutter is used to access the device's location services.
3. FCM stands for **Firebase Cloud Messaging.** 5
4. **mockito** package in Flutter helps you create Mock Dependencies of classes while testing.

Explain Accessibility in Flutter apps:
Accessibility in Flutter ensures that apps are usable by everyone, including people with disabilities. Flutter provides features like screen reader support (TalkBack/VoiceOver), large fonts, high-contrast themes, and semantic labels for widgets.
Example: Semantics(label: 'Submit Button', child: 5
ElevatedButton(onPressed: () {}, child: Text('Submit')),);
-This helps assistive technologies describe the app's UI properly.

Give steps to integrate camera in Flutter app: 5
Steps to integrate camera functionality:
1. Add camera and path_provider packages in pubspec.yaml.
2. Import the packages in your Dart file.
3. Initialize the camera using availableCameras() and create a CameraController.
4. Display a camera preview using CameraPreview(controller).
5. Capture an image using controller.takePicture().
These steps allow a Flutter app to access and use the device's camera for photos or videos.

Explain Push Notifications in Flutter using Firebase:
Push notifications in Flutter are implemented using Firebase Cloud Messaging (FCM). It allows the app to receive messages even when it's not running.
Steps:1. Add firebase_core and firebase_messaging packages in pubspec.yaml.
2.Configure Firebase in the Flutter app using the Firebase Console.
3.Initialize Firebase in the main() function. 5
4.Use FirebaseMessaging.onMessage.listen() to handle notifications when the app is open.
Example:
FirebaseMessaging.onMessage.listen((message) { print(message.notification?.title); });
- FCM helps send alerts, updates, or messages to users in real time.

Explain Testing and Debugging in Flutter:
Flutter provides built-in tools for testing and 5
debugging apps efficiently.
Testing types:
Unit Testing: Tests individual functions or logic.
Widget Testing: Tests UI components in isolation.
Integration Testing: Tests the complete app flow.
Debugging tools:
Flutter DevTools: Inspect widgets, memory, and performance.
Hot Reload: Quickly apply code changes without restarting the app.
Debug Console: View logs and errors.
Example for a unit test:
test('adds two numbers', () { expect(2 + 3, 5); });
These tools help developers find bugs early and improve app quality.

Explain Animation Techniques – Flare and Lottie:
Flutter supports advanced animations using Flare (Rive) and Lottie. **Flare (Rive):**Used to create and control vector animations that respond to user actions. -**Example**: RiveAnimation.asset('assets/intro.riv'); -It's interactive, smooth, and lightweight. **Lottie**:
Displays prebuilt JSON-based animations made in After Effects. -**Example**: Lottie.asset('assets/loading.json');
- It's best for loading icons or small effects. -Both help make the app look modern and dynamic without heavy code.