Week 1

1- what is flutter?

flutter is an open-source UI framework developed by **Google** or an UI toolkit for building natively compiled applications for [mobile(Android , IOS) , desktop (Windows, Mac , Linux) and Web] with a single codebase (cross platform) .

Also it uses **Dart** programming language (which is fast , flexible & support OOP)

*Now why use flutter ? or what is flutter's **features** ?

1-**Beautiful , modern & elegant UI** : via set of rich and customizable widgets.

2- **Hot Reload:** allows developers to instantly see changes in the app without restarting it, boosting productivity.

3-**High Performance:** Flutter uses its own **Skia Graphics Engine (2D)** , delivering near-native performance

4-**Strong Community**: Active support and a wide range of available packages.

* However nothing can survive from **Drowbacks** so Flutter is **a great choice** for cross-platform development, but if the app requires **highly optimized performance, small app size, or deep native integrations**, we might need to consider **native development**

⇨ More discussion about the main feature for flutter (from my point of view) :

2- How can Flutter build apps for different platform using one codebase?

Flutter is a cross platform ,uses a single codebase (which means write once deploy many),

Through it's **unique architecture** and **rendering engine…**

1- Flutter's Architecture : (layered architecture)

a. Dart Code:

- The developer writes the app logic and UI using **Dart**.
- Dart compiles into native machine code or JavaScript, depending on the platform.

b. Flutter Framework (Widgets):

- The **Flutter framework** provides a rich set of **widgets** for building UIs.
- These widgets are **platform-agnostic** (meaning they don't depend on platform-specific UI components but instead describe the UI in a declarative way).

c. Flutter Engine:

- The **Flutter engine** uses the **Skia graphics library** to render widgets directly on the screen.
- This engine is responsible for translating the Dart code into platform-specific behavior and rendering.

d. Platform Channels:

- For accessing platform-specific features (like camera, GPS, etc.), Flutter uses **platform channels** to communicate with native code written in Java/Kotlin (Android) or Swift/Objective-C (iOS).

2- Rendering Process:

Flutter does not use native UI components of the underlying platforms. Instead, it renders everything using its own graphics engine (**Skia**), ensuring a consistent look and feel across platforms. The rendering process works as follows:

- Flutter draws **every pixel** on the screen, bypassing native widgets.
- This approach gives complete control over the UI, making it possible to maintain a consistent appearance across platforms.
- Flutter achieves **60 FPS** (frames per second) performance by using efficient rendering pipelines.

3- Compilation Process:

Flutter follows different compilation strategies **depending on the target platform**:

a. **For Mobile** Platforms (Android & iOS):

- Dart code is compiled into native **ARM or x86** machine code using Ahead-of-Time (AOT) compilation.
- This allows the app to run with near-native performance.

b. **For Web:**

- Flutter compiles Dart code to JavaScript using **Dart2JS**.
- The app runs inside a **browser** with a Flutter-rendered UI using **HTML and Canvas**.

c**. For Desktop**:

- Flutter compiles Dart code into a **native binary** (using **AOT** compilation), similar to mobile platforms.
- The app interacts with the underlying OS directly via the Flutter engine.

4- **Platform-Specific Adaptations:** it allows for platform-specific customizations when needed:

- **Platform-Specific Widgets:** Flutter provides widgets that adapt to the platform, such as Cupertino widgets for iOS and Material widgets for Android.
- **Conditional Code Execution:** we can use Dart's platform **detection** (Platform.isAndroid, Platform.isIOS,… etc.) to run different code on different platforms.
- **Plugins and Packages:** Flutter offers a large ecosystem of **plugins** that provide access to platform-specific features (camera, GPS, and sensors).

=> Benefits : (fast development, consistency, cost-effective & high performans**)**

For more details : Flutter architectural overview | Flutter

3- What is meant by: directory, package, library, framework, SDK, IDE?

a- **Directory** (folder) : is a container used to organize files and subdirectories on a computer (ex. **lib** directory includes main dart code)

b- **Package** : collection of related files(including code, configuration files, and metadata, that provides specific functionality).(ex. **provider** or **http**).

c- **Library** : is a reusable collection of code that provides specific functions, often grouped logically.(ex**. dart:math** for mathematical operations)

d- **framework**: is a pre-built structure that provides tools, libraries & best practices to simplify development**.**(ex. flutter is aframework)

e- **SDK(Software Development Kit):** is a collection of tools, libraries, and documentation required for developing applications for a specific platform**.**
(ex. flutter SDK containing Dart compiler, build tools, and libraries for Flutter development)

f- **IDE(Integrated Development Environment):** is a software application that provides tools like a code editor, debugger, and compiler to streamline development
(ex. VS Code, Android Studio, and IntelliJ IDEA).

4- What is a Widget? Give as many examples as you can.

A widget : is the fundamental building block of a user interface. **Everything in Flutter is a widget**, including elements like text, buttons, images and the layout structures. Widgets define the structure, appearance, and behavior of the UI components. They are immutable (cannot change once created) and are organized in a widget tree that determines how elements are laid out and displayed.

Widgets can be classified into two main types:

- **StatelessWidget**: A widget that does not change its state once it is built.
- **StatefulWidget:** A widget that can change dynamically based on user interactions or data updates.

Examples:

1- **Container Widget**(a versatile widget used for layout, decoration & positioning)
2- **Scaffold Widget (**provides a basic structure with app bar ,body & other elements)
3- **AppBar Widget** (displays a toolbar at the top of the screen)
4- **Column Widget (**arranges its children vertically)
5- **Row Widget (**arrenges its children horizonally)
6- **Button (ElevatedButton) Widget (** clickable button with an elevation effect)
7- **Text Widget (**displays a piece of text on the screen).
8- **Image Widget (**displays an image from an assest or network)
9- **listView Widget** (displays a scrollable list of widgits)
10- **icon Widget** (displays an icon from a set of predefined icons)

5- What is state? and What is the difference between stateless and stateful widgets?

State: refers to the data that can change during the lifetime of a widget & determines how it looks and behaves at a given moment.

feature	Stateless widgets	Stateful Widget
Definition	A widget that does **not** change once it is built.(immutable)	A widget that can **change** dynamically based on user interactions or data updates.
State management	**No** internal state, UI remains constant.	**Has** an internal state that can change over time.
Usage	Used for **static** UI elements	Used for **dynamic** UI element
Examples	Text, Icon , Image	Checkbox , Slider , AnimatedContainer
Rebuild	Rebuilds only when the parent widget changes(in widget tree) It has no state so **no recall** for **build()** function .	**Can rebuild** independently when its state changes. we call **build()** function for the first time , then when dynamic action occurs (like clicking a button) a call for **setState()** function occurs then increment counter and a **recall** happens **for build()** function.