Flutter Development

What is Flutter?

Ans: Flutter is an **open-source development framework developed** by Google to build mobile, web, and desktop applications from a single codebase. It's an SDK(Software Development Kit) using Dart language.

What makes Flutter unique?

Flutter is different from other frameworks because it **neither uses WebView nor the OEM widgets that are shipped with the device**. Instead, it uses its own **high-performance rendering engine to draw widgets**. It also implements most of its systems such as animation, gesture, and widgets in the Dart programming language that allows developers to read, change, replace, or remove things easily. It gives excellent control to the developers over the system.

What is Dart?

Ans: Dart is a general-purpose, object-oriented programming language with C-style syntax. It is open-source and developed by Google in 2011. The purpose of Dart programming is to create **frontend user interfaces for the web and mobile apps**.
It is an important language for creating Flutter apps. The Dart language can be compiled into both **AOT (Ahead-of-Time) and JIT (Just-in-Time).**
Just-in-Time (JIT) is a type of compilation that c**ompiles your app in the browser at runtime**.
Ahead-of-Time (AOT) is a type of compilation **that compiles your app at build time**

Advantages of Flutter?.
Ans:

Open-Source: Flutter is a free and **open-source framework for developing** mobile applications.

Cross-platform Development: This feature allows **Flutter to write the code once, maintain it, and can run on different platforms**. It saves the time, effort, and money of the developers.

Faster Development: The performance of the **Flutter application is fast.** Flutter **compiles the application** by using **the arm C/C++ library that makes it closer to machine code** and gives the app a better native performance.

Live and Hot Reloading: It makes the app development process extremely fast. This feature allows us to change or update the code as soon as the alterations are made.

UI Focus: It has an **excellent user interface** because it uses a **design-centric widget, high-development tools, advanced APIs**, and many more features.

limitations of Flutter?

Ans:
Larger Release Size: Flutter frustrates developers when release size is not as per their expectations(always larger).

Learning a new language dart which is not a very popular language.
New technology.
The modern framework tries to separate logic and UI as much as possible but, in Flutter, **user interface and logic are intermixed.** it's difficult to separate.

It provides **very limited access to SDK libraries**. It means a **developer does not have a lot of functionalities to create a mobile application**. Such types of functionalities need to be developed by the Flutter developers themselves.

What is Flutter Widgets?

Ans: Widgets are basically **user interface components** used to create the user interface of the application. helps us to create a user interface of any complexity.
In *Flutter*, **the application is itself a widget**. The application is the top-level widget and its UI is built using one or more children (widgets), which again build using its children's widgets.

A Flutter app is always considered as a **tree of widgets**. Whenever you are **going to code for building anything in Flutter, it will be inside a widget**. Widgets describe how your app view should look with their current configuration and state.

Design Specific Widgets

The Flutter framework has two sets of widgets that conform to specific design languages. These are **Material Design for Android application** and **Cupertino Style for IOS application.**

GestureDector is an **invisible widget**, which includes tapping, dragging, and scaling the interaction of its child widget.

Types of Widget: We can split the Flutter widget into two categories:

1. **Visible(Output and Input) :Text,button(flat button or raised button),image,icon etc**
2. **Invisible (Layout and Control): column,rows,center,padding,scaffold,stack**

Flutter Scaffold: design visual layout structure

Container: parent widget that can contain multiple child widgets

Row And Columns: align children horizontally and vertically according to our needs.

Text: display a string of text with a single line in our application.

TextField : text input widget

Flutter Button: provides a user to trigger an event

Flutter Stack: *to overlap multiple widgets into a single screen* and render them from bottom to top.

Flutter Forms: Flutter provides a Form widget to create a form.

Flutter Icons: Icon Widget to create icons in our applications.

Flutter Drawer: create a sliding left menu layout

Flutter List:

Flutter Toast Notification: Flutter Toast is also called a **Toast Notification message.** It is a very small message which mainly **pops up at the bottom of the device screen**. It will **disappear on its own after completing** the time provided by the developers.

Flutter CheckBox: to choose multiple options from several selections.

Flutter Radio Button: Boolean value. It allows the user to choose only one option.

Progress Bar: show the progress of a task such as downloading, uploading, installation, file transfer, etc

SnackBar: a lightweight message that briefly informs the user when certain actions occur.

Flutter Charts: represent the data in a graphical way

Bottom Navigation Bar: contain multiple items such as text labels, icons, or both

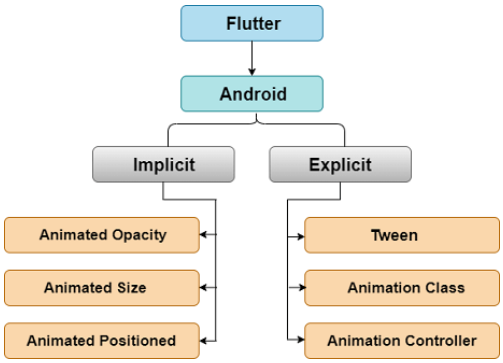
Flutter Calendar: table_calendar to show the calendar in our app.

Flutter Animation: Animations **enhance user experiences** and make the **applications more interactive.two main categories, which are given below:**

- **Tween Animation**
- **Physics-based Animation**

Tween Animation: It is the short form of **in-betweening**. In a tween animation, it is required to **define the start and endpoint of animation.** It means the animation begins with the **start value,** then goes through a **series of intermediate values, and finally reaches the end value**. It also provides the **timeline and curve,** which defines the time and speed of the transition

Physics-Based Animation: It is a type of animation which allows you to make an app interaction feel **realistic and interactive.** It simulates the **real-world animation/movement**, such as you want to animate a **widget like spring, falling, or swinging with gravity.** Thus, it is an animation that animates in response **to user input/movement**.



Hero Animation: A hero animation is a type of animation where an element of one screen **flies** to a new screen when the app goes to the next page.

What is a State?

A **state is a piece of information that can be read when the widget is built and might change or modified over the lifetime of the app**. If you want to change your widget, you need to update the state object, which can be done by using the setState() function available for Stateful widgets. The **setState()** function allows us to set the properties of the state **object** that triggers a redraw of the UI.

Flutter **is declarative.** It means Flutter builds its UI by **reflecting the current state of your app.**

What do you understand by the stateful and stateless widget?

Ans:

Stateful: A Stateful widget has **state information**. It is referred to as **dynamic** because it can **change the inner data during the widget's lifetime**.
A widget that allows us to refresh the screen is called a Stateful widget.
The examples of the Stateful widget are **Checkbox, Radio, Slider, InkWell, Form, and TextField.**
This widget does not have a **build() method**. It has the **createState()** method, which returns a **class that extends the Flutters State Class**.

Stateless: The Stateless widget does **not have any state information**. It remains **static** throughout its lifecycle.
The examples of the Stateless widget are Text, Row, Column, Container, etc.
If the screen or widget contains **static content**, it should be a **Stateless widget**,
but if you want to change the content, it needs to be a Stateful widget.

Android Studio

It is **free, open-source, and the fastest tool** used to build **an app on every type of android device.** It provides a **complete experience for developing** Flutter applications such as **code completion, navigation, syntax highlighting, refactoring, widget editing assists, and run & debug support, etc**. The main purpose of the android studio is to accelerate the **development process and build high-quality apps** for all android devices. It allows the **developer to fix certain code issues automatically.**

Hot Reload vs Hot Restart?

Ans:

Hot Reload:

- The hot reload feature quickly **compiles the newly added code in our file and sends the code to the Dart Virtual Machine**. After updating the Code Dart Virtual Machine, update **the app *UI* with widgets**.
- Flutter hot reload features work with the combination of the Small r key on the command prompt or Terminal.
- It helps to build UI, add new features, fix bugs, and make app development fast.
- Hot Reload takes **less time** than Hot restart.
- There is also a **drawback in Hot Reload**, If you are using *States* in your application then **Hot Reload preservers the *States*** so they will not update on Hot Reload or set to their default values

Hot Restart:

- It mainly works with **States value**.
- It allows developers to get a **fully compiled application** because it **destroys the preserved State values** and **sets them to their defaults**. On every Hot Restart, our app widget tree is completely rebuilt with **the new typed code**.

- Hot Restart takes **much more time** than Hot reload.

Different Build modes in flutter?

Ans: The Flutter tooling supports three modes while compiling the application. These compilation modes can be chosen depending on where we are in the development cycle. The names of the modes are:

- Debug
- Profile
- Release

Navigation and Routing: Navigation and routing are some of the core concepts of all mobile applications, which allow the user to **move between different pages**. In Flutter, the **screens and pages** are known as **routes,** navigating to different pages defines the workflow of the application, and **the way to handle the navigation** is known as routing.

Flutter provides a basic routing class **MaterialPageRoute** and two methods **Navigator. push()** and **Navigator. pop()**

Flutter Packages: A package is a namespace that contains a **group of similar types of classes, interfaces, and sub-packages.**
In Flutter, Dart o**rganizes and shares a set of functionality through a package**. Flutter always supports **shared packages**, which are contributed by **other developers to the Flutter and Dart ecosystem.** The packages allow us to **build the app without having to develop everything from scratch.**

Types of Package: According to the functionality, we can categorize the package into two types:

1. **Dart Package**
2. **Plugin Package**

Dart Package: It is a **general package**, which is written in the **dart language, such as a path package**. This package can be used in both the environment, whether it is a web or mobile platform.

Plugin Package: It is a specialized Dart package that includes **an API written in Dart code and depends on the Flutter framework.** It can be combined with a platform-specific implementation for an underlying platform **such as Android (using Java or Kotlin), and iOS (using Objective C or Swift)**. An example of this package is the **battery and image picker plugin package.**

Flutter REST API: **most of the apps use remote data using APIs.**Flutter

provides an **HTTP package to use HTTP resources.** The HTTP package uses **await and async features and provides many high-level methods** such as r**ead, get, post, put, head, and delete methods for sending and receiving data from remote locations**. These methods simplify the development of REST-based mobile applications.

Flutter Database: A database is an **organized collection of data**, which supports

the s**torage and manipulation of data** and is accessed electronically from a computer system. Flutter provides many packages to work with the database. The most used and popular packages are:

- **SQLite database:** It allows users to **access and manipulate SQLite databases**.
- **Firebase database:** It will enable you to **access and manipulate the cloud database.**

SQLite Database: SQLite is a popular **database software library** that

provides a **relational database management system for local/client storage**. It is a **lightweight and time-tested database engine** and contains **features like self-contained, server-less, zero-configuration, transactional SQL database engine.**

Flutter SDK does **not support SQLite directly**. Instead, it provides a plugin **SQLite,** which **performs all operations on the database similar to the SQLite library.**

Firebase Database:

Firebase is **a Backend-as-a-Service**, and it is a **real-time database** that is basically designed for mobile applications. **Firebase (a NoSQLjSON database)** is a real-time database that **allows storing a list of objects in the form of a tree.**

Firebase has **three main services**, i.e., a **real-time database, user authentication, and hosting.** We can use these services with the help of the **Firebase iOS SDK to create apps without writing any server code**.

Why use Firebase?

- Firebase manages **real-time data in the database**. So, it **easily and quickly exchanges the data to and from the database**. Hence, for developing mobile apps such as **live streaming, chat messaging, etc.**, we can use Firebase.
- Firebase **allows syncing real-time data across all devices** - iOS, Android, and Web - without refreshing the screen.

- Firebase provides integration to **Google Advertising, AdMob, Data Studio, BigQuery DoubleClick, Play Store, and Slack to develop** our apps with efficient and accurate management and maintenance.
- **Everything from databases, analytics to crash report**s is included in Firebase. So, the app development team can stay focused on improving the user experience.
- Firebase applications can be **deployed over a secured connection to the firebase server.**
- Firebase offers a **simple control dashboard.**
- It offers a number of **useful services to choose from**.

Cons

- Firebase is not widely used, or battle-tested, for enterprises.
- It has very limited querying and indexing.
- It provides no aggregation.
- It has no map-reduce functionality.
- It cannot query or list users or stored files.

The **Firebase Realtime Database is a NoSQL database** from which we can store and **sync the data between our users in real-time**. It is a **big JSON object** which the developers can manage in **real-time**. By using a **single API**, the Firebase database provides the application with the current value of the data and updates to that data. Real-time syncing makes it easy for our users to access their data from any device, be it web or mobile. **When our users go offline**, the Real-time Database SDKs use **local cache on the device for serving and storing changes**. The local data is **automatically synchronized when the device comes online.**

In the **Firebase Real-time database,** data is stored **as JSON objects**. We can think of the **database as a cloud-hosted JSON tree**. There are **no tables and records, which means it is a NoSQL database**. When we add data to the JSON tree, **it becomes a node** in the existing **JSON structure with an associated key.** We can provide our own keys, such as user IDs or names, or they can be provided to us using the **push() function**.

Flutter Vs React Native

Concept	Flutter	React Native
Develop By	It was first introduced by Google.	It was first introduced by Facebook.
Release	May 2017	June 2015
Programming Language	It uses **Dart** language to create a mobile app.	It uses **JavaScript** to create mobile apps.
Architecture	Flutter uses **Business Logic Component (BLoC) architecture.**	React Native uses **Flux and Redux architecture**. Flux was created by Facebook, whereas Redux is the preferred choice among the community.
User Interface	It uses custom widgets to build the UI of the app.	It uses native UI controllers to create the UI of the app.
Documentation	Flutter documentation **is good, organized, and more informative**. We can get everything that we want to be written in one place.	React native documentation is **user-friendly but disorganized.**

Performance	The performance of **the Flutter application is fast**. Flutter compiles **the application by using the arm C/C++ library** that makes **it closer to machine code and gives the app a better native performance.**	The performance of the React **Native app is slow in comparison to the Flutter app**. Here, sometimes developers face issues **while running the hybrid application architecture**.
Testing	Flutter provides a **very rich set of testing features**. This feature allows the developer to **perform unit testing, integration testing, and widget testing**.	React Native uses **third-party tools that are available for testing the app**.
Community Support	It has less community support as compared to React Native.	It has very strong community support where the questions and issues can be solved quickly.
Hot Reload	Supported	Supported

Tell Me about your project?

Inventors is basically an education startup where I worked as a **Frontend Mobile App developer.**in this, I worked in a team and we made an education app called IVara. This education app was developed for the student and teacher community to provide a platform for the **conduction of quizzes, delivery of lectures, and seamless interaction** among students, teachers, and parents where parents can also see the progress of their child. We made this application using a **framework known as flutter.**

What was your role in this project?

Here I worked as a **frontend develope**r so my task was to make the **user interface of this app**.
In this, I was given a **task to make splash screens login signup screens**, and our app was divided into three sections **student, teachers, and parents** so my task was to make **UI for parents as well as students section**. like in the parent's section I have to make screens where parents can see **teachers' lists, their child's attendance as well as scores in exams** and can contact individual teachers using chat features.

What difficulties did you face during the project?

non-tech difficulties

1. conflicts between teammates ideas

How did you resolve that:
i)**our leader asked everyone's opinion** and then finally made **an overall good decision for all teammates.**
ii)Low engagement of teams in the initial phase

- how did all teammates work efficiently then?
we all understand our roles and work accordingly, to make a better app for the hackathon

tech difficulties

1. **Video calling feature issue**
While **using agora SDK**, we were having screen frame issues, **i.e frames were running faster and were loaded,** more than required. due to which, buffer flow caused the mobile to hang.

- **how did u resolve that**
We upgraded **all the dependencies** and the agora SDK with the latest version made all the previous functions to new ones and r**ematched them with the latest documentation.**

2. We have made a dual login for teacher and student, this whole was in a single app, we could make two apps, for a more efficient backend. like zomato, swiggy also made their own apps this way(one for consumer, one for producer)

Why did you make this app?

Basically, when I entered college I have various doubts related to college clubs and other things..and in starting days we can't contact seniors due to some ragging restrictions and...it was difficult for us to gain knowledge in particular tech or club or anything....and also having some common grp on WhatsApp its very difficult to distinguish between seniors juniors aur our batchmate if we don't know them...so to solve this issue we came up with this idea we will help students to distinguish their students on the basis of years they are studying tech or clubs...and can have interaction

Does it make any changes to another life?

Ans: yes ..we haven't published our app yet but we have noticed among ourselves how easily we can interact with our seniors mentor aur even alumni and discuss doubts in a particular interest or can take help and do the same for juniors.

Future Goals

We are planning to publish our app on the play store after certain changes or the removal of bugs so students from various colleges from various places in India can interact among their seniors, mentors, professors, etc.

reason-
We have not used any architecture, like MVC, or Bloc, due to which some parts of the UI become interdependent with others.

like, user A, just after logging in, if he chooses a student, and fills in some details and pressed the button.
At the same time, use B, search that user, there will be some details that haven't been updated.
so if we had used the MVC pattern, Modal, view, Controllers, we could have solved this problem.

We partially implemented this architecture, just the modal and views part.

For the backend, which backend service did you use? And why?

Basically, **we use firebase,** which is basically an **API, to manage servers or server-side code.** Much **prebuilt code like updating, creating document snapshots are really easy with that**.

As we can easily create, read, update, delete operations easily over the firebase rest API.
We used a NoSQL database for our mobile application. as I mentioned already, the firebase cloud store

Why did you use a NoSQL database, why not SQL?

as our data was not having bundles of entries, we were restricted to our college stuff.
so, **less data is really easy to organize using NoSQL**(unstructured database).

but for large databases or entries, we have to use SQL, for organized way and fast queries.

SQL provides us ACID properties, which definitely helps the database redundancies, anomalies.

Ek bar ye dekh lo, main to SQL, NoSQL, ye chize hein.

tech
1. databases hi hai main.
2. Which API is used? (firebase)
3. What are APIs?
4. Which type of data format an API usually provides?
ans: JSON or XML
4.1 Which one will you prefer?

ans: JSON, as it is easy to read and easy to use.

non-tech
1. team me tumhara role.
2. how did u resolve team conflicts