

Q1

- a) Explain the key features and advantages of using flutter for mobile app development?
- Single database for multiple program write one codebase for both Android and IOS, reducing development effort and maintenance.
1. Hot Reload instantly see changes in the app without restarting making development faster and more interactive.
  2. Fast Performance Uses the Dart language and a compiled approach for smooth and high performance app.
  3. Open source & strong community support - Backed by google and a large developer community ensuring continuous improvement and resources.

### Advantages

1. Faster development time: Hot reload and single codebase reduces development time significantly.
2. Cost effective: Since the same code runs on both Android and IOS, businesses save on development and maintenance cost.

3 Reduced Performance issues: The app runs natively without relying on intermediate bridges like in react native reducing lag.

b) Discuss how the flutter framework differs from traditional approaches and why it has gained popularity in the development community.

→ Single codebase vs Separate codebase

Traditional approach: Developers need to write a separate code for Android and iOS. Flutter uses a single dart based codebase for both platforms reducing development time and effort.

2 Rendering Engine vs native UI components

Traditional approach relies on platform native UI components which can lead to inconsistencies and performance issues. Flutter uses the skia rendering engine to draw everything from scratch ensuring a consistent UI across devices.

Why flutter has gained popularity

1. Fast development with hot reload  
Developers can now instantly see UI changes without restarting the app making the operations process much quicker.
2. Cross platform Efficiency: Business save time and resources by maintaining a single code base for multiple platform.



3. Constant VT across devices since flutter does not rely on native components, the UI looks and behaves the same different OS version
4. Improved performance: AOT compilation and direct access to GPU rendering ensures smooth animations and high performance

Q1) Describe the concept of the widget tree in flutter. Explain how widget composition is used to build complex user interface

→ Widget tree in flutter

In flutter, the widget tree is the fundamental structure that represents the UI of an application. It is a hierarchical arrangement of widgets where each widget defines a part of the user interface. Flutter UI are built using widgets which can be Stateless or Stateful. The widget tree determines how the UI is rendered and updated when changes occur. Widget composition refers to building complex UIs by combining smaller, reusable widgets.

Example:

```
class ProfileCard extends StatelessWidget {  
  final String name,  
  final String imageUrl,
```

Profile card (I required this name, required this image)  
@ override

Widget build (Build content content) {  
return Card (

child column (

children [

Image network (image url)

Sized Box (height: 10)

Text (name, style: TextStyle (fontSize: 20,

})

})

## Benefits

1. Reusability Small widgets can be reused in different parts of the app
  2. Maintainability Breaking UI into smaller widgets makes it easier to debug and update
  3. Performance Flutter efficiently rebuild only the necessary part of the widget tree.
- b) Providing examples of commonly used widgets and their roles in creating a widget tree
1. Structural Widget.  
These widgets acts as the foundation for building the UI
  - Material app: The root widget of a flutter app that provides essential configuration
  - Scaffold: Provides a basic layout structure,

including an app bar, body floating configuration

Example:

Material App C

home: Scaffold C

app.Bar: AppBar C Title: Text C "Flutter  
widget tree")

body: container C

padding: Edge (Inset all (16.0))

child: text ("Hello, Flutter!")

),

),

),

Input & interaction widgets

Text field: accepts text input from user

Elevated Button: A button with elevation

Gesture button: Detects gestures like taps swipes  
and long press.

Ex. Column C

Children [

Textfield (decoration, InputDecoration (label text: "Enter name")

Elevated Button C

OnPressed: () {

print ("Button Pressed");

},

child: text ("Submit")

)

];



Q3)

a) Discuss the importance state management in flutter applications

→ In flutter, state refers to data that can change during the lifetime of an application. This includes:

- User inputs
- UI changes
- Network changes
- Animation states

There are two types of states

1. Ephemeral state: small, UI specific state that doesn't affect the whole app.

2. App wide states - Data stored across multiple widgets importance of state management.

- Efficient UI updates: Flutter UI: it rebuild wherever state changes. Efficient state management ensures that only necessary widgets are updated, improving performance.
- Code maintainability & scaling: Managing state

b) Compare and contrast the different state-management approaches available in flutter such as setState, Provider and Riverpod. Provide scenarios where each approach is suitable.

setState : Local state

Pros: simple build in easy to use

Cons: Not scalable unnecessary re-renders

Best use cases: Small UI updates

Provider : App wide state

Pros: Light weight recommended by flutter, efficient

Cons: Boiler plate code for nested provider

Best use case: Large apps needing global state

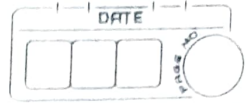
Riverpod App : Wide state

Pros - Eliminates Provider limitations

Cons - Requires learning new concepts

Scenarios for each approach

- Use setState when managing simple UI elements with in a single widget like toggling dark mode in a setting screen.
- Use provider when sharing state across multiple widgets such as managing user authentication & theme changes.
- Use Riverpod when building a complex scalable app with global state management like an ecommerce app with cart management.



firebar with  
cloud firestore  
Run: flutter pub get

- Step 4

Configure firebase for android  
for android

1. Open android / build.gradle and ensure the full classpath 'com.google.gms:google-services:4.3.10'
2. Open android / app / build.gradle and add it at the bottom apply plugin: 'com.google.gms:google-services'.

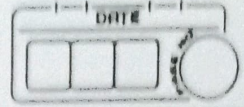
• step 5

Initialize Firebase in flutter

```
void main() async {
```

## # Benefits of Using Firebase





4. Firebase Cloud Messaging  
Enables push notifications and managing known users  
Eg: Firebase Messaging instance . subscribeToTopic ("news")

5. Firebase hosting  
Deploys and serves web applications securely with automatic ssl

Data Synchronization in firebase  
Firebase ensures real time data synchronization across multiple devices and platform using ~~firebase~~ Firestore and Realtime Database.

1. Cloud firebase Sync Mechanism  
Uses real time listeners to update UI instantly when data changes

Ex: Firebase Firestore . instance collection ('users').  
snapshots (). listen ('Snapshot') {  
for (Var doc in snapshot) {  
    print (doc [name]);  
}  
}

2. Realtime Database Sync Mechanism  
Uses persistent Websocket connections for live updates.