VARUN GUPTA
D15B - 18
DATE:

AH 04

Q1. Explain key features & advantages of using flutter for mobile app development

Ans

A] Single codebase for multiple platforms. One codebase for both Android and iOS, reducing development effort.

i) Hot reload - instantly see changes in app without restarting making development faster
ii) Fast performance - use dart language and a compiled approach for smooth & high performance apps.
iii) Open source & strong community support - backed by Google and a large developer community, ensuring continous improvement

Advantages:
1) Faster development time : Hot reload & single codebase reduce development time
2) cost effective since code runs of both Andriod and iOs, business save on development & maintainence.
3) Reduce performance issues. The app runs natively without relying on intermediate bridge like in react native.

B] Discuss how flutter framwork differs from traditional approach and why it has gained popularity.

Ans:
Single codebase VS Separate codebase
Traditional Approach : Developers need to write separate code for Android & iOS
Flutter uses a single dart based codebase for platform reducing time & effort.

2. Rendering Engine vs Native UI
Traditional approach: relies on platform native UI component which can lead to inconsestency.
Flutter uses the ski rendering engine to draw environment from scratch ensuring consistent UI across devices.

Why Flutter gained popularity?
Ans.
i) Faster development with Hot Restart can instantly UI changes without restarting app making development easier.
ii) Cross platform efficiency : business save time & resources by maintaining single codebase for multiple platform.
iii) Consistent UI aceross devices since ~~features~~ flutter does not rely on native components.
iv) Improved performance : No compilation and direct access to GPU rendering ensuring smooth animation.

Q2. Describe the concept of widget tree in flutter. Explain how widget composition is used to build complex UI.

Ans.
Widget Tree in Flutter
i) It is the fundamental structing UI of application.
ii) It is hierarchical arrangement of widget can be stateless or stately.
iii) Widget tree determine how UI is rendered and updated when changes occur.

Widget Composition in Flutter

i) It refers to building complex UI ~~of~~ by ~~widget composition~~ combining smaller reusable widget.

ii) Instead of creating large monolithic UI component, Flutter encourages breaking the UI into smaller widgets to be reused.

eg.

```
    class Profilecard extends stateless widget {
          final string name;
          final string component;
      Profilecard ( { required this.name, req. this.img url } );
      @ override
      widget build ( Build context context ) {
             return card (
                   child column (
                       children. [
                   Image network (image_url')
                   . SizedBox (height : 10)
```

Benefits:

1) Reusability - small widgets can be reused index parts of app.

2) Maintainability - Breaking UI into smaller widget makes it easier to debug & update.

3) Performance - Flutter efficiently rebuilds only the necessary parts of widget tree.

Firebase Authentication
Enables secure authientication using email/password phone no. & third party providers like Google, facebook & Apple.

2] Cloud Firestore - store and sync data in real time across device support structure data queries and offline access

eg.
```
Firebase Firestore instance collection ('user').add({
    'name' : 'JohnDoe':
    'email' : 'JohnDoe @gmailcom'
  });
```

3] Realtime Database
A realtime JSON database that automatically updates data across devices.
eg.
```
Database Refrence ref = Firebase. Database instance ref.
let ( {"text": "Hello, Firebase"})
```

4] Firebase cloud messaging (FCM)
Enables push notification and messaging between users
ex. Fire messaging. instance subscribe ToTopic ("news")

5] Firebase Hashing
Deploys and serves webapp securely with automatic

→ Data Synchronisation in Firebase
Firebase ensures real time data synchorisation across multiple devices and platforms using firestore and creating database.

1) Cloud firestore sync mechanism.
Use realtime listners to update UI instantly when data changes.
ex. Firebase. Firestore instance. collections (" users"). snapshot()

```
    y. for (vardoc. snapshot. docs)
           print (doc [name]);
```

→ Runtime Database Sync. Mechanism.
uses persistent websocket connection you live. updates.
ex.

```
Database Reference refs = firebase. database. finstance.
                                        ref("Message").
   ref on value listen ((event){
        print( event. snapshot. value)
        });
```

→ Offline Data sync.
Firestore caches data locally & sync changes when the developer onlin.
ex. Firebase firestore. instanc. setting. setting.

→ Cloud function for automated updates.
Automates backend logic to trigger when data changes.