Att OS

Name:- Nidhi Rijhwani
Class:- D15B Roll No: 47
MPL Assignment No: 01

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

→ Flutter is a popular open-source UI toolkit developed by Google for building natively compiled applications for mobile (iOS & Android) web and desktop from a single codebase.

⇒ Key features of flutter:-

1. Single codebase: Write once, run on multiple platform (iOS, Android web, desktop).
2. Dart Programming language:- Uses Dart, which is optimized for fast performance and ahead-of-time (AOT) compilation.
3. Hot Reload: Instantly reflects changes in the app without restarting, making development faster and more efficient.
4. Rich Widget Library: Provides a vast collection of customizable widgets that support Material design and cupertino styles for a native look and feel.

⇒ Advantages of using flutter:

1. Faster development Time: Hot reload and a single codebase reduce development effort and time.

2. cost effective :- Since developers write one codeba
for multiple platforms, it reduces costs associa
with maintaining separate teams for ios & Android.

3. Consistent UI :- flutter renders everything
using its own engine, ensuring a uniform
look across devices.

Q1. b) Discuss how the flutter framework differs from
traditional approaches ? and why it has
gained popularity in the developer community?

→ flutter uses a single codebase for multiple
platforms, unlike traditional native development that
requires separate code for IOS (swift) & Android (Kotlin)
It doesnot rely on platform - specific UI components
but instead render everything using its own
skia graphic engine, ensuring consistency.
Unlike React Native, which uses a Javascript
bridge, flutter compiles directly to native
ARM code, offering better performance. Its
hot reload feature allows developer to see
changes instantly, making development faster
& more efficiency.

flutter has gained popularity due to its faster
development cost efficiency, & cross platform
support. Business prefer it as redues
development apps. It customizable widget
system. ensures a smooth native like
experience.

2. a) Describe the concept of the widget tree in flutter Explain the widget composition is used to build complex UI.

→ In flutter, everything is a widget (button, text, lay out etc). These widgets are arranged in a hierarchical structure known as the widget tree. The widget tree determines the UI.

- Widget composition to build complex UI:-
- flutter encourages a composition-based approach rather than inheritance.
- Instead of creating large, monolithic widget, developer build small, reusable widget, that are combined to form complex UIs.
- ex. A column widget can hold multiple Text & button widget, creating a structured layout.

Q2. b) Provide ex of commonly used widgets & their roles in creating a widget tree.

→ 1) Structural widget;
- scaffold : Provide basic structure of a screen.
- container : Used for layout styling.
- column & Row : Used for vertical & horizont -al layout.

2) Interactive widget
- Text field : for user input
- Eleveted Button : Clickable buttons.

3). Styling widget,

• Padding: Adds spacing around widget.
• Align, centre: Adjust alignment.

4) List & scrollable widget,
• Listview: scrollable list
• Gridview: Provide / display items in Grid.

ex:- Simple widget Tree

```
scaffold (
 appBar: AppBar (title: Text ("Flutter App")),
  body: column (
    children: [
      Text ("welcome to flutter")
      ElevatedButton (onPressed: () { }, child:
         Text ("Click Me"))
    ],
  ),
);
```

Q3 a) Discuss the importance of state management in flutter application.

→ Importance of state management in flutter application state management refers to handling dynamic data that changes overtime.
- In flutter, the UI rebuilds when the state changes, ensuring the app remains interactive & responsive proper State management helps in performance. optimization, code maintainability & better UI behaviour.

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

→ Comparision of state management Approaches in flutter
Approach Description Suitable Scenarius setstate
Basic state Management by calling setstate() to update UI small apps, simple UI updates (eg toggling a switch)
Provider user Inherited widget to efficiently manage state across the widget tree. Medium sized apps needing global state sharing (eg, user authentication)
Riverpod More scalable than provider with improved dependency injection & state handling. large, complex apps requiring modular & scalable state management (eg. e-commerce apps).

Q4 a) Explain the process of integrating firebase with a flutter application.
Discuss the benefits of using firebase as a backend solution.

→ Integrating firebase with flutter & its benefits:-
Integration Process:
Setup firebase Console:

Create a firebase project
Register the App for Android & ios
Download & app google-services.json (Android) or google service-Info.plist (ios)
Install firebase dependencies:

```
yaml
dependencies:
firebase-core : latest-version
firebase-auth : latest-version
cloud_firebase : latest-version
```

Initialize firebase in flutter

```
dart
void main() async {
   widget flutter Binding.ensureInitialized();
   await firebase.initializeApp();
   runApp(myApp());
}
```

Benefits :-
No need to manage servers (Backend-as-a-service)
Provide authentication database & cloud function
scalable & cost-effective.

Q4. b) Highlight the firebase services commonly used in flutter development & provide brief overview of how data synchronization is achieved.

→ commonly used firebase services in flutter & data synchronization service functionality.
firebase authentication NoSQL database for real-time data syncing. firebase storage upload & manage files (images, videos) cloud messaging push notifications, firebase Analytics app usage analytics.

Data synchronization in firebase :
firebase allows real-time data syncing using snapshot listner.

ex, of real-time listener in firebase :
dart

```
firebase firestore • instance • collection ('message').
   snapshots().
   listner ((snapshot) {
        for (var doc in snapshot.docs){
            print (doc. data ());
        }
   });
```