Gungun · Kamlesh · Panjabi
D15A 64

MAD ASSIGNMENT NO 1

Q1]

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

→ 1] Single codebase for multiple platforms. Write on codebase for both Android and ios reducing development effect and maintainance.

2] Hot Reload Instantly see changes in the app without restarting making development faster and more interactive

3] Fast Performance : use the dart language and a compiled approach for smooth and high performance app

4] 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 sengle codebase reduce development time significantly

2] cost effective : Since the same code run and both Android and ios, business save on development and Maintainc cost

3] Reduce performance issues : The app run natively without relying on intermediate bridges like in react native reducing lag

b] Discuss how the flutter framework differs from traditional approach and why it has gained popularity in the developed community.

→ 1] Singe codebase vs separate codebase

• Traditional Approach : Developers needs to write applicable

hierarchial arrangement of widgets with each widgets design a part of the user interface. Flutters UI is entirely built ~~with~~ using width which can be stateless or. The width tree determines how the UI is rendered and updated when changes occurs

widget component in flutter:

widget component refers to building complex UI by combining smaller, reuseable widgets. Instead of creating large monolithic UI components, Flutter encourages the UI into smaller manageble widget that can be reused and nested with each other

Example: class profilecard extends stateless widget {
final string nam ;
final string image url;
profile card ({required this name, required this image url}).
@ override
widget build (Build context context){
return card (
child column (
children [
image.network (image.url)
Sized Box (height: 10)
Text (name, style: Textstyle (front size: 20, fontweight: Fontweight.bold,))
]
)
}.

Benefits of weight composition

1] Reusability: Small widget can be reused in different part of the app

2] Maintainability: Breaking UI into smaller widget Make it easy to debug and update

Separate code for Android (Java/ koffin) and ios (swif
• Flutter uses a single dart based codebase for both
platform reducing development time and effect

2] Rendering Engine Vs Native UI components.
Traditional Approach : Relies on platform native UI
compent which can lead to inconsistencies and perfo
issues flutter uses the skia rendering engine to d
everything from scatch ensuring a consistent UI
across devices.
Why flutter has gained popularity.
1] Faster development with hot reload. Developers c
instantly see UI changes without restarting the a
Making the iteration process much quicker.
2] cross platform efficiency : Busines save time and
resources by maintaining a single codebase for mul
platforms
3] consistent UI across devices. Since flutter does not re
native component the UI looks and behave the
across different os versions
4] Improved performance : Aot compitation and direct
to GPU rendering ensure smooth animation and high per

Q2]

a] Describe the concept of the widget free in futter. Exp
how widge composition is used to build complex us
interfaces.

→ widget free in flutter
In flutter the widget tree is the fondamental str
that represent the UI of an application. It is a

```
onPressed : () {
  Print ("Button Pressed")
  },
  child : Text ("submit")
  }
  }
  };
```

3] Display of styling widget

- Text - display text on the screen
- Image - shows image from assets network or Memory
- Icon - Display icons.
- card - A Material design card with rounded corners and elevation

```
Ex : column (
        children:[
    Text ("welcome to flutter!", style : Textstyle (font size : 24;
                                    fontweight : font weight - bold)
IMage .network ("https // flutter .dest / image / flutter - logo -
                                                sharing .png")
  ] ;
) ;
```

Discuss the important of state Management in flutter applications.

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

- user input
- UI changes
- network changes
- Animation state

SDS Page No.
Date / /

3] Performance : Flutter efficiently rebuild only the necessary part of the widget tree

6] Provide example of commonly used widget and their role in creating a widget tree.

→ 1] structural widget :
These widget act as the foundation for building the UI
• Material app : The root widget of the both flutter app that provides essential configuration
• Scaffold : Provides a basic layout structure, including an app ber body floating action button etc
• container : A versatile widget used for styling, padding margin and background customization

```
Ex : MaterialApp (
        Home : Scaffold (
        appBer : AppBar (title Text ("flutter widget Tree"
        body : container (
        Padding : EdgeInsets . all (16.0)
        child . Text ("Hello, Flutter!"),
        ),
        ),
        );
```

2] Input & Interaction widgets
Textfield - Accept text input from users.
Elevation Button . A Button with elevation
gesture Detector : Detectors gesture like taps, swipes and l
presses.

```
Ex : column (
        children . [
        Textfield (decoration . Input decoration ( label Text : "Enter
        Elevated Button (
```

SDS Page No.
Date / /

There are two type of states.
1] Ephemeral state : small, UI specific state that doesn't affect the whole app
2] App wide states : Data stored across multiple wid
importance of state Management
- Efficient UI updates : Flutter's UI is rebuilt whenever sta changes. Effecient state Management ensure that only ne widgets are update, improving performance
- code Maintainability & Scability : Managing style proper Make the code Modular. redable and scalable fi larger application.
- Data consistency & synchronization : proper state Ma ensure that data remains consistent across different screen and widgets

b] Compare and contract the different stateManagement approaches avaliable in flutter such as setstate, provider and Riverpod. Provide scenarios where approach is suitable

→ Setstate. local state
Pros - Simple built in case to use
cons - Not scalable causes unnecessary re-render
Best use cases - small UI update (eg- toggle, switch,

Provider - App wide state
Pros - light weight, recommended by flutter efficient
cons - Boilerplate code for nested providers.
Best use case - Medium scale app (eg. Authentication, M API data)

SDS | Page No.
Date | |

Riverpod App : wide state (More scalable than provider)
Pros - Eliminates Providers limitations improves performance
cons - Require learning new concepts.
Best use cases - large app needing global state (eg shopping cart, user sessions

Secnarios for each Approach.
use setstate when Managing simple UI element within a single widget like toggling dark Mode in a setting screen
use provider when staring state across Multiple widgets such as Managing user authentication or theme changes
use Riverpod when building a scable app with global state Management, like an ecommerce app with cart Management.

Explain the process of integrating firebase with a flutter application. Discuss the benefits of using firebase as a backend solution.
Firebase provides a powerful backend solution for flutter application offering services like authentication real time database, cloud functions, storage and more
Steps to integrate Firebase with flutter

Step 1 : create a firebase project
- go to firebase console
- click on "Add project" and enter a project name
- configure google analytics if needed, then click create

Step 2: Register the flutter app with firebase
- In the firebase project dashboard click "add App" a
select Android or ios based on your platform
- for Android : Enter the android package name and
download the google services is on file and place
it in android /app/.
- for ios.
Enter the ios Bundle identifier
Download the google service-info-plist file and place
it in a ios/runner

Step 3 : Install firebase dependence
Add firebase dependence in pubspec.yaml
firebase core.
firebase auth
cloud firestore
Run. flutter pub get

Step 4 : configure Firebase for android & ios for Andro
1] open android / build.gradle and ensure the f
classpath 'com.google.gms:google-services:4.3.10'
2] open android / app / build.gradle and add at t
bottom apply plugin: 'com.google'.gms.google-service

Step 5 : Initialize Firebase in flutter
void main () async {
widget Flutter Binding.ensure Initialized ();
await Firebase initialize app ();
runApp (My App());

SDS | Page No.
Date / /

Benefits of using firebase
Firebase is a Backend as a service (Baas) that simplifies backend development for flutter app. Here are some key benefits

1] Easy to set up & scale
No need to manage backend infrastructure
Scales automatically based as usage.

2] Authentication
Provides email / password, google, facebook and phone authentication.
Seamless integration with firebase Authentication

3] cloud storage
Secure file storage for image, video and documents.

4] Push Notification (firebase cloud Messaging) send real-time notification to users across different platform.

Highlight the firebase services commonly used in flutter development and provide a brief overview of how data synchronization is achieved.
Firebase provides a suite of backend services that simplify flutter app development

1] Firebase Authentication
Enables secure authentication using email / password, phone number and third party provides like google, facebook and apple

2] cloud firestore
Store and syncs data in real time across devices
support structure data queries and offline access
eg : firebase.firestore.instance.collection('users').add({
'name': 'John Doe',
'email': 'johndoe@example.com.
});

SDS | Page No.
Date | |

3] Realtime Database
A realtime Json-based database that automically
data across devices.
ex : Database Reference ref = Firebase Database.instance.ref(
ref.set ({"test": "Hello, Firebase!"});

4] Firebase cloud Messaging (FCM)
Enable push notifications and Messaging between
Ex. Firebase Messaging.instances.subscribeTo Topic("news

5] Firebase Analytics.
Track user interactions and app performance
Ex : Firebase Analytics analytics.firebase Analytics.instance
analytics.logEvent (name: "button clicked", parameters:
{"button": "subscribe"});

6] Firebase Hosting
Deploys and serves web applications security wi
automatic

- Data Synchronization in Firebase.
Firebase ensures real-time data sync synchronizatio
across multiple devices and platform using firesto
and realtime Database
1] cloud Firestore Sync Mechanism
uses realtime listeners to update UI instantly when
changes
Ex : Firebase Firestore.instance.collection ('users').snapshots
listen ((snapshot) {
for (var doc in snapshot.docs) {
print (doc ['name']
}
});

SDS Page No.
Date / /

2] Realtime Database sync Mechanism:
uses persistent web socket connections for live updates
Ex: Database Reference ref = firebase Database.instance.ref ("Message")
ref.onValue.listen ((event) {
print (event.snapshot.value);
});

3] offline Data sync.
firestore caches data locally and syncs changes when the devices is online
Ex:- Firebase firestore.instances.setting.settings (Persistence Enabled true);

4] cloud function for automated updated.
Automates backend logic to trigger updates when data changes