Abhay Gupta
D15B
16

DATE 04

MAD Assignment - 1

a) Explain features, advantages of using flutter for mobile app development.

1. Single for multiple platforms: write one codebase for both android and IOS, reducing development effort and maintenance

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

3. Fast performance: Uses Dart language and a compiled approach smooth and high performance apps.

4. Open source & sharing community support, Enforced by google and large developer community. ensuring continuous improvement and resources.

Advantages:-

1. Faster Development Time: Hot reload and single codebase reduce development time significantly

2. Cost Effective: The code runs both android and IOS, business save on development and maintain and cost

b). Discuss how flutter framework work diff from the

traditional approach and why gained popularity in development community.

1. Diff from traditional approaches:

Rendering Engine: Traditional framework rely on pla
specific UI

Declarative UI: UI Updates are defined as part
of state change rather than imperative

Performance: Flutter apps have better performance
than react Native as avoid JS bridge

2. why flutter popular in Dev Community:

Rapid development with hot Reload
Strong support from google and the community
Growing number of plugins and Libraries
Flexibility in UI customization
Support for web mobile and desktop

Q2) a) widget tree and widget Composition in flutter

ans In flutter, the widget tree is fundamental structure
that represent the UI of an application. It's a hierar
arrangements of widget where each widget defines a
part of user Interface

widget Composition in flutter refers to building

complex UI by combining smaller, reusable widgets. Instead of creating large, monolithic components flutter encourages breaking the UI is rendered and updated when changes occur

Example: class ProfileCard extends statelessWidget {

```
final String name;
    final String imageURL;

ProfileCard (req.this.name, req this.imgurl});
@override
widget build (Build context) {
    return Card (
        child. Column (
            children. [
                Image. network
                    (Image URL)
} }}
```

Benefits of widgets composition :-
Reusability small widgets can be reused in diff parts of the app

Maintainability Breaking UI into smaller widgets makes it easier to debug and update

Performance: flutter efficiently rebuild only necessary parts

b) Provide examples of commonly used widgets and roles in creating widget tree

1. Structural widgets
These widgets acts as the foundation for buil the UI

- Scaffold Provides basic layout structure, includi or app bar, body floating action button etc

Eg:- MaterialApp(
home: Scaffold(
appBar: AppBar(title:
('Flutter Widget Tree')),
body: Container(
padding: EdgeInsets
));

2. Input & Interaction widgets

ElevationButton: A button with elevation gestures like taps, swipes and long presses

Eg:- Column(
children[
TextField(decoration: InputD
(labelText: "Ente
ElevatedButton(
onPressed: (){
Print("Button Pr
});

3. Display & Styling Widgets

- Text - displays text on the screen
- Image - shows images from assets network or memory
- Icon - Displays Icons
- Card - A material design card with rounded corners and elevation

Eg:- Column(
children[
Text("welcome to flutter", style.
TextStyle(Fontsize: 24,
FontWeight.bold)),
Image.network("https://flutter.dev
images/flutter-logo-sharing")),
];
);

a) In flutter, state refers to data that can change during the lifetime of an application
This includes:
- User input
- UI changes
- Network changes
- Animation states

There are two types of states
1. Ephemral state: small UI, specific state that doesn't affect the whole app
2. App wide state: Data shared across multiple

widgets Importance of state management ensures that only necessary widgets are updated improving perfor

Data Consistency and Synchronization
proper state management ensures that data remains consistent across different screens and widgets

b) set state - Local state
Pros - simple built in easier to use
Cons - Not scalable causes unnecessary re-rend

Best use Cases - small UI updates (eg. toggle s counter)

Riverpod App - wide state CMore scalable than provid
pros - diminates Providers limitation, improves performances
Cons - Requires learning new concepts
Best use case - large apps needing global state

Scenarios for each approach

→ Use set state when managing simple UI Elements with a single widget like toggling dark mode in a setting screen

→ Use provider when sharing state across multiple widgets such as managing user authentication or theme changes, like an ecommerce app with cart management

SDS | Page No.
Date / /

a) Explain process of integrating firebase with a flutter application

Ans 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

Step 2: Register the flutter app with firebase
- In the firebase project dashboard click "Add App" and select android or IOS
- For Android: Enter the android package name and download the google service json file and place it in android app

Step 3: Install firebase dependencies
Add firebase dependencies in pubpes.yaml
firebase core
firebase with cloud firestore
Run flutter project

Step 5: Initializing firebase in flutter

SDS | Page No. | Date

```
void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp();
  runApp(MyApp());
}
```

→ Benefits of using firebase:

1. Fast to setup & scale
2. Authentication
3. Cloud Storage
4. Push notification

b) Firebase provides a suite of backend services th simplify flutter app development

1. Firebase authentication:
   Enable secure authentication using E-mail / password, phone no. and third party providers like google, facebook and google and apple
2. Cloud firebase
   Stores and syncs data in real time ac devices support structured data, queries and offline access
3. Realtime Database:
   A real-time, JSON-based database that auto updates data across databases.