Aditya Raj
D15B
01

MPL Assignment - 1.

04

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

→ i) Single codebase for multiple platform. write one codebase for development faster and more interactive.
ii) fast performance: uses the past language and a complied approach for smooth and high performance apps.
iii) Hot Reload :- Instantly see the changes to the app without restart.
iv) Open source and strong community support.
v) Faster development Time :- Hot reload and single codebase reduces development time.
vi) Cost effective :- since the code runs on both android and ios, businesses save on development cost.
vii) Reduced performance issues :- The apps runs natively without relying on anything.

Q. Discuss how the flutter framework differ from traditional approaches and why it has gained popularity.

→ i) Single codebase v/s seprate codebase
ii) Traditional approach: developers need to write seprate code for android (Java / Kotlin) and ios (swift).
iii) flutter : uses a single dart based codebase for both platform reducing development time and effect.

b. Provide example of commonly used widgets and their roles in creating a widget tree.

→ ① Structural widget :-

- MaterialApp :- The root widget of flutter app.
- Scaffold :- provides a basic layout-structure.
- Container :- A variable widget for styling, padding, margin and background.

e.g :-
```
MaterialApp(
  home: Scaffold(
    appBar: AppBar(title: Text("APP")),
    body: Container(
      padding: EdgeInsetsAll(16.0),
      child: Text("Hello"),
    ),
  ),
)
```

② input and interaction widgets.

TextField :- Accepts text input from users.
Elevation Button :- A button with elevation.
GestureDetector :- detects gestures like taps, swipes and long presses.

e.g
```
Column(
  children: [
    TextField(decoration: InputDecoration(
      labelText: "Name"),
```

Q1) Rendering engine vs Native UI components

Traditional approach :- relies on platform, native UI components which can lead to inconsistencies.
Flutter uses the rendering engine to draw everything.

Why flutter gained popularity :-

i) faster development with Hot reload which makes the process faster.
ii) Cross platform efficiency : Buisnesses can save time by maintaining a single codebase.
iii) Consistency across devices, the UI looks and behave consistently across all platforms.
iv) Easy integration with Backend :- works well with Firebase, REST APIs, GraphQL.

Q2) Describe the concept of widget tree in flutter. Explain composition is used to build complex user interface.

→

widget tree is a fundamental structure that represents the UI of the application. It is a hierarchical arrangement of where each widget defines a part of the user interface, maintains the rendering and updation of the UI.

```
ElevatedButton(
    onPressed: () {
      print("Button pressed");
    },
    child: Text("Submit"),
  ),
);
}
```

(3) display and styling widgets.

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

e.g. :-

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

step 3 :- install firebase dependencies.
add dependencies in pubspec.yaml
firebase core.
firebase auth
cloud firestore.

step 4 :- configure firebase for android and ios.

step 5 :- initialize firebase in flutter.

```
void main () async {
  widget flutterBinding. ensureInitialized ();
  await Firebase. initializeApp ();
  runApp ( MyApp ());
```

benefits of using firebase :-

1. easy to setup and scale
2. provides proper authentication
3. cloud storage facility.
4. cloud messaging and notifications

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

→ firebase provide a suite of backend services that simplify flutter app development.

1. firebase authentication.
enables secure authentication using email/password, phone number and third party services like google, facebook.

2. Cloud Fire
Stores and sync data in real time across devices.

3. Realtime database : automatically updates data across devices.

4. cloud messaging (FCM) : enables push notifications and messaging.

5. firebase Analytics.
Track user interactions and app performance.

6. Firebase Hosting
deploys and serves web applications securely with automatic SL.

provider - App wide state
Pros - lightweight, recommended by flutter.
cons - Boiler plate, code for nested provided
best use cases - medium scale apps

Riverpod :- App wide state (more scalable than provider)
- Eliminates providers.
- requires learning new concepts.
best use cases: large apps including global state.

Q4) 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 applications offering services like authentication, real time databases, cloud functions, storage and more.

steps to integrate firebase with flutter.

Step - 1 :- Create a firebase project.
configure google analytics if needed.

Step 2 : register the flutter app with firebase.
configure the registration according to your platform.

Q.3) Discuss the importance of state management to flutter application

→ 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 Ephimeral state - ~~data~~ shared ~~across multiple widgets~~ small, UI specific state that doesn't affect the whole app.

② App wide states - data shared across multiple widgets importance.

Importance of state management:-

- Efficient UI updates
- Code maintainability and scalability
- data consistency and synchronization.

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

→ setstate - local state

Pros - simple built in case to use.

Cons - Not scalable causes unnecessary re-renders.

Best use cases - small UI updates

(eg toggle switch, counter).

widget composition in flutter :- refers to the building complex UIs by combining UI components, reusable widgets. Instead of creating large, monolithic components, flutter encourages breaking the UI into small manageable widget.

example :-

```
class profile extends statelessWidget {
final string name,
final string imageURL,
profile card ({required this.name});
@override
widget build (BuildContext Context) {
return card (
child: column(
Text (name, style: TextStyle
(fontSize: 20, fontWeight: ));
}
```

benefits of widgets composition.

i) Reusability :- small widgets can be reused in different parts of the app.

ii) Maintainability :- easy debugging and maintainance.

iii) performance :- Improves efficiency by rendering.

Data synchronization in firebase :- ensures real-time synchronization across multiple devices.

1. cloud firestore synchronization.

e.g firebase.firestore.instance.collection("users").snapshots.
listen((snapshot) {

for (var doc in snapshot.docs){
print(doc['name']);
}
};

2. Realtime database sync Mechanism
uses persistent websocket connections for live updates.
ref.onValue.listen((event){
print(event.snapshot.value);
});

3. offline data sync
firestore caches data locally and sync changes when device online.

4. cloud functions for automated updates
automates backend logic to trigger updates when data changes.