Tarunkumar sharma
D15B
Roll no - 52
MAD Assignment

Q1
a) Explain the key features and advantages of using flutter for mobile app development?
→ Single database for multiple program write one codebase for both Android and IOS, reducing development effiert and maintence.
2 Hot Reload instantly see changes in the app without restarting making development faster and more interactive.
3 Fast Performace Uses 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 continous improvement and resources

Advantages
1 Faster development time: Hot reload and single codebase reduces development time significa
2 Cost effective: Since the same runs code run on both android and ios, business save o development and maintance cost :

3 Reduced Performance issues : The apps runs natively without relying on intermediate bridges like in react native reducing lag.

b) Discuss how the flutter framework differ from traditional approaches and why it has gained popularity in the development community

→ Single codebase VS Seprate codebase
Traditional approach: Developers need to write seprate code for Android and ios
Flutter uses a single dart based codebase for both platforms reducing development time and effort

2 Rendering Engine VS native UI components
Traditional Approach Relies on platform, native UI components which can lead to inconsistens and performance issues flutter. Uses the skia rendering engine to draw everything from scratch; ensuring a consistent UI across devices

Why flutter has gained popularity
1. Faster development with hot reload
Developers can see instantly see UI changes without restarting the app making the operations process much quicker
2. Cross platform Efficiency : Buisness save time and resources by maintaining a single code base for multiple platform

3. Constant UI Across devices since flutter does not rely on native components, the UI looks and behaves the same across different OS version

4. Improved performance : AOT compilation and direct access to GPU rendering ensures smooth animations and high performance

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

→ Widget tree in flutter

In flutter, the widget tree is the fundamental structure that represent the UI of an application. It is a hierarchical arrangement of widgets where each widget defines a part of the user interface. Flutter UI are built using widgets which can be stateless or stateful
The widget tree determines how the UI is rendered and updated when changes occur
Widget Composition refers to building complex UI's by combining smaller, reusable widgets

Example:

```
class ProfileCard extends StatelessWidget {
    final String name,
    final String imageUrl,
```

```
ProfileCard ({required this.name, required this.image})
  @override
  Widget build (BuildContext context) {
    return Card (
      child Column (
        Children [
          Image network (image url)
            Sized Box (height : 10)
            Text (name, style : TextStyle (fontsize: 20,
        ]
      )
    }
```

Benifits

1. Reusability: Small widgets can be reused in different parts of the app
2. Maintainability: Breaking UI into smaller widgets makes it easier to debug and update
3. Performance: Flutter efficiently rebuild only the necessary part of the widget tree.

b) Providing examples of commonly used widgets and their roles in creating a widget tree

1. Structural Widget

These widgets acts as the foundation for building the UI

- Material app: The root widget of a flutter app that provides essential configuration
- Scaffold: Provides a basic layout structure,

including an app bar, body floating configuration

Example:

```
Material App (
    home: Scaffold (
        app. Bar. AppBar (title. Text ("Flutter
                                widget tree")
    body. container (
        padding: Edge (Insert all (16.0)
        child. text ("Hello, Flutter!")
    ),
    ),
    ),
```

Input & interaction Widgets.

Text field. accepts text input from user

Elevated Button. A button with elevation

Restore button. Detects gestures like taps swipes and long press.

Ex. Column (

```
    Children[
    Textfield (decoration, InputDecoration (labelText,"
                                Enter name")

    Elevated Button (
        OnPressed: () {
            print ("Button Pressed"),
        },
        child. text ("Submit")
        )
    ]);
```

Q3)

a Discuss the importance state managment in flutter applications

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

- User inputs
- UI changes
- Network changes
- Animation states

There are two types of states

1 Ephemeral state: small, UI specific state that doesn't affect the whole app.

2 App wide states - Data stored across multiple widgets importance of state managment.

- Efficient UI updates: Flutter UI rebuild whenever state changes. Effied state managment ensures that only nessary widgets are updated, improving performance.
- code maintainability & scaling: Managing state

b) Compare and contrast the different state-managment approaches available in flutter such as setstate, Provider and Riverpod Provide scenario where each approach is suitab

setState : Local state
Pros: simple built in easy to use
Cons: Not scalable unnessary re-renders
Best use Cases: Small UI Updates

Provider: App wide state
Pros: Light weigh recommended by flutter, effiect
Cons: Boiler plate code for nested providers
Best use case: Large apps needing global state

Riverpod App: Wide State
Pros - Eliminates Provider limitations
Cons - Requires learing new concepts

Scenaroes for each approach
- UseState when managing simple UI elements with in a single widget like toggling dark mode in a setting screen.
- Use provider when sharing state across multiple widgets such as managing user authentication of theme changes
- Uses Riverpod when building a complex scalable app with global State managment like an ecommerce app with cart managment

firebase auth
cloud firestore
Run: flutter pub get

- Step 4:
Configure firebase for android
for android
1. Open android/build.gradle and ensure the following classpath 'com.google.gms:google-services:4.3.10'
2. Open android/app/build.gradle and add at the bottom apply plugin: 'com.google.gms.google-services'

- Step 5
Initialize Firebase in flutter

```
Void main() async {
    Widget flutterBinding ensureInitialized();
    await firebase initialize App();
    Run App(MyApp());
```

Benifits of Using firebase
1. Easy to setup & scale
No need to manage backend infrastructure
Scales automatically based on usage
2. Authentication
Provides email/password, google, facebook and phone authentication
Seamless integration with the firebase Authentication

4. Firebase cloud Messaging
Enables push notifications and messaging between users
Eg: FirebaseMessaging.instance.subscribeToTopic("news")

5. Firebase hosting
Deploys and serves web applications securely with automatic SSL

Data Synchronization in firebase
Firebase ensures real time data synchronization across multiple devices and platform using firestore and real time Database

1. Cloud firebase Sync Mechanism
Uses real time listners to update UI instantly when data changes

Ex: Firebase Firestore.instance.collection('users').
snapshots().listen((snapshot){
for(var doc in snapshot){
print(doc[name]);
}
});

2. Realtime Database Sync Mechanism
Uses persistent Websocket Connections for live updates.