1

Jeetu Mamtora
Div-D15B Rollno-30.

(04) (A)

Assignment-01.

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

→ ~~Fe~~ Flutter is a popular open-source UI toolkit developed by Google for building natively compiled applications for mobile (ios & Android) web & desktop from a single code base.

Key features of flutter-
1) Single Codebase- write once, run on multiple platform (ios, Android, web, desktop).

2) ~~Data~~ Dart Programming- Uses Dart, which is optimised for fast performance. and ahead-of-time (AOT) compil
3) Hot Reload- Instantly reflects changes in the app without ut restarting, making development faster.
4) Rich widget Library- Provides a vast collection of customization widgets that supports Material design & cupertino styles for a native look & feel.

Advantages of using flutter-
1) Faster Development Time- Hot reload & a single code-base reduce development. effort & time.
2) cost Effective- Since developers write once codebase for multiple platforms, it reduces costs associated with maintaining separate teams for ios & android.
3) consistent UI- Flutter- renders everything using its own engin

1) 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 seperate code for IOS (swift) & Android (kotlin). It does not 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 efficient.

Flutter has gained popularity due to its faster development cost efficient & cross platform support Business prefer it as it reduces development time & cross while delivering high performance apps. Its 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, layouts, etc).
These widgets are arranged in a hierarchial 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. eg A column widget can hold multiple Text & Button widget, creating a structured ~~inf~~ layout.

2)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 & Horizontal layout.

2) Interactive widget.
- Textfield - for user input.
- Eleveted Button - clickable buttons.

3) Styling widget.
- Padding - Adds Spacing around widget.
- Allign, centre - Adjust alignment.

4) List & Scrollable widget.
- Li~~s~~stview - Scrollable widget.
- 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(ompressed: ( ) { }, Child: Text ("Click
       me")
  ];
);
);
```

FOR EDUCATIONAL USE

Q3)a) Discuss the importance of State management in Flutter application-

→ Importance of State Management in flutter Application State Management refers to ~~having~~ handling dynamic data that changes overtime. InFlutter, the UI rebuilds when the State changes, ensuring the app remains interactive & responsive proper State management helps in performance optimization, code ~~[illegible]~~ maintainability & better UI behaviour.

Q3)b) Compare & 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 Scenarios SetState Basic State Management by calling SetState() to update UI Smallapps, Simple UI updates (eg toggling a Switch) Provider uses 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).

4)a) Explain the process of integrating firebase with a flutter application.
Discuss the benefits of using firebase as a backend Soln.
Integrating firebase with flutter & its benefits.

→ Integration process:
Setup firebase console:

Create a firebase project
Register the App for Android & ios.
Download & add google-services.json (Android) or Google Service-info.plist (ios)
install Firebase Dependencies.

yaml

```
dependencies-
    firebase_core: latest_version
    firebase_auth: latest-version
    cloud_firestore: latest_version
```

initialize firebase in flutter.

dart

```
void main() async{
  widgetsflutter Binding.ensureInitialized();
  await firebase.initializeApp();
   runApp(MyApp());
}
```

Benefits -
No need to manage servers (Backend-as-a-Service) Provide authentication, database & cloud function, scalable & costeffective.

4)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 user-sign-in (Email, Google, Facebook)
Cloud FireStore 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 FireStore -
Firestore allows realtime data Syncing using snapsho listener.

Ex - of realtime listener in FireStore.
dart.

```
Firebase firestore.instance.collection('message').snapshots().
listen((snapshot){
      for (doc in snapshot.docs){
        Print(doc.dart());
      }
});
```