Shrusti Chintawar
D15B- 9

Page No.
Date 4

MPL - Assignment 1

A+

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

Ans:- 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 codebase

i. key features of flutter:

1. Singe codebase: write once, run on multiple platform (ios, android, web, desktop)
2. Dart programming language: uses dart, which is optimized for fast performance and ahead-of-time (AOT) compilation
3. Hot reload: Instantly reflects changes in the app without restarting, making development faster & more efficient
4. Rich widget library: provides a vast collection of customizable widgets that support material design & cupertino styles for a native look & feel

ii. Advantages of using flutter

1) faster development Time: hot reload & a single code base reduce development effort & time

2) cost effective: Since developers write one codebase for multiple platforms, it reduces costs associated with maintaing

separate teams for ios & android

3) Consistent UI: flutter renders everything using its own engine, ensuring a uniform look across devices.

Q 15. Discuss how flutter framework differs from traditional approaches & why it has gained popularity in the developer community?

Ans: flutter uses a single codebase for multiple platforms, unlike traditional native development that requires separate code for ios (swift) & android (kotlin). It does not rely on platform-specific UI components but instead renders 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 efficiency & cross platform support. Business prefer it as it reduces development time & costs while delivering high performance apps. Its customizable widget system ensures a smooth native like experience

Q 2 a) Describe the concept of the widget tree in flutter. Explain the widget composition is used to build complex UI.

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

Ans: In flutter everything is a widget ~~tree~~ in (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.

ex. A column widget can hold multiple Text & button widget, creating a structured widget.

Q2.b) provide ~~an~~ of commonly used widgets & their role in creating a widget tree

Ans: i) Structural widget

- Scaffold: provide basic structure of a screen
- container: used for layout styling
- column & row: used for vertical & horizontal layo-
- ut

ii) Interactive widget

- Textfield: for user input.
- Elevated button: clickable buttons

iii) Styling widget
- padding: adds spacing around widget
- align, center: adjust alignment

iv) List & scrollable widget
- listview: scrollable widget
- gridview: provide / display items in grid

eg: Simple widget Tree

```
Scaffold (
  appBar: AppBar (title: Text ("flutter App")),
body: column (
  children: [
  Text ("Welcome to flutter!"),
  ElevatedButton (on pressed: () {}, child: Text ("
  click Me"))
  ],
  );
  );
```

Q 30) Discuss the importance of state management in flutter application

Ans: Importance of State management in flutter application
State management refers to handling dynamic data that changes over time
In flutter, the UI rebuilds when the state changes ensuring the app remains interactive & responsive
proper state management helps in performance optimization, code maintainability & better UI behaviour

Date

Q3b Compare and contrast the different state managem-ent in flutter approaches available in flutter, such as setstate, provider & Riverpod, provide scenarios when each approach is suitable

Ans: comparison of state management approaches in flutter approach description suitable scenarios setstate basic state management by calling setstate() to update UI. small apps, simple UI updates (eg., to-ggling a switch) provider user inherited widget to efficiently manage state across the widget tree Med-ium sized apps needing global state sharing (eg, user authentication) RiverpodMore scalable than provider with improved dependency injection & st-ate handling. large, complex apps requiring modu-lar & scalable state management (eg. e-commerce apps)

Q4a) Explain the process of integrating firebase with a flutter application
Discuss the benifits of using firebase as a backend solution

Ans: Integrating firebase with flutter & its benifits-
Ans: 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 {
WidgetsFlutterBinding.ensureInitialized();
await Firebase.initializeApp();
runApp(MyApp());
}

Benefits:
No need to manage servers (backend-as-a service) provide authentication, database & cloud functn scalable & cost-effective

Q4b. Highlight the firebase services commonly used in flutter development & provide brief overview of how data synchronizatn is achieved

Ans: Commonly used firebase services in flutter & data synchronizatn service functionality
firebase Authentication users-sign in (Gmail, google facebook) cloud firestore NoSQL database for real-time data syncing service fr firebase storage & manage files (images, videos) cloud messaging push notifications, firebase analytics app usage analytics

Page No.
Date

Data synchronizatn in firebase:

firestore allows real time data syncing using snashot listner

eg. for real-time listener in firestore dart

```
Firebase.firestore.instance.collectn('message').snapshots().
listen((snapshot){
for(var doc in snapshot.docs){
        print(doc.data());
    }
});
```