Aaqueeb Pinjari
D15B/56

NAME:_____ STD.:_____ DIV.:_____

DATE :
PAGE : 1

Assignment - 1

Q.1.a) Explain the key features and advantages of using flutter for mobile app development.

→ ① Single code base :-

Develop for IOS and android from a unifide codebase, reducing development time and efforts.

② Hot reload :-

Real time code changes without restarting the run process, enhancing development efficiency.

③ Rich widget library :-

Pre-designed, costomizable widgets for consistant and visually appealing user interface.

④ High performance :-

Flutter compiles to native ARM code & uses the skia graphics engine, ensuring smooth performance.

⑤ Cost effective :-

Reduce development cost with single code base & efficient development process.

b) Discuss how the flutter framework differs from traditional approaches & why it has gained popularity in the development community.

→ ① Dart language :-

Flutter employs dart, a language specific to framework different from platform specific languages used in traditional approach.

(c) Efficient and Time saving :-
Flutter reduces development time for IOS & android by enabling single code base. Also reduce development time by implementing code reusability.

(d) Consistant UI across the platforms :-
Flutter ensures a uniform user interface on IOS & android.

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

→ In flutter, the widget tree is a hierarchial representation of user interface components where each node corresponds to a widget defining the structure & appearance of the UI widget serve as a fundamental building block, ranging from basic elements like buttons & text to move to complex structure. Widget composition is a core concept in flutter, allowing development to build intricate user interfaces through the assembly of simple & reusable widgets. This process involves combining, nesting & configuring widgets to create module components. Developers start with foundational widgets & progressively compose them into more sophisticated structures.

The hierarchial arrangement of widgets in the tree mirrors, the layout & composition of the UI widgets can be nested, allowing for the creation of complex interfaces.

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

→ ① Dynamic user interface :-

State management is critical for handling dynamic changes in user interfaces. Wether it is updating UI elements in response to user interactions as reflecting changes in data effective structure management ensures that the UI remains responsive & reflects the current application state.

② Code reusability :-

Well managed state enables the creation of modular & reusable components. In flutter, where widgets can be composed and reused, effective state management ensures that these components can be easily integrated into different parts of application, promting a PRY codebase.

③ Cross screen communication :-

State management facilitate communication between different screens or components of an application, allowing them to share & synchronize data.

b) Compare and contrast the different state management approaches available in flutter such as set state, provider & riverpod provide scenarios where each approach is suitable.

→ ① Set state :-

The set state method is a built-in mechanism in flutter for managing the internal state of stateful widget.

Scenarios :-

Set state is suitable for small to moderatly complex UI's where state changes are localized to specific widget & dont to be shared across the entire application.

② Provider :-

The provider package is a popular & light weight state management solution in flutter. It follows the provider pattern & is based on inherited widget.

Scenarios :- Provider is suitable for managing state within specific parts of widgets tree, creating a scoped and efficient solution.

③ Riverpod :-

Riverpod is an advanced state management library and a successor to provider. It provides a broader set of features & its designed to be more modular & testable.

Complex applications :- Riverpod is suitable for large & complex application where more structured & testable state management approach is needed.

Q.4. a) Explain the process of integrating firebase with flutter application. Discuss the benifits of using fire base as a backend solution.

→ ~~Create Firebase Project~~

① Start by creating a project on fire base console & configure your app, Add firebase to flutter project.

② In your flutter project, add the necessary dependencies by updating pubspec.yaml file.

```
yaml :
dependencies:
firebase_core : ^ Letest-version
firebase-auth : ^ Letest_version'
cloud_firestore : ^ Letest-version
```

③ Run flutter pub get to fetch the dependencies.

④ Initialize firebase in your flutter app by calling firebase initialize App(), in the main() method :- dart :-

⑤ Import package :- ' Firebase-core.dart ';

```
void main() async {
    widgets flutter Binding ensure initialized();
    await firebase initialize App();
    run App (MyApp());
}
```

⑥ Use firebase service like authentication ~~firebase~~ firestone in flutter app by importing the relevant packages and initializing them using firebase project credentials dart.

```
import 'Package.firebase_auth/firebase_auth.dart';
import 'Package.cloud_firestore/cloud_firestore.dart';

// Authentication
  firebaseAuth auth = firebaseAuth.instance;
  user ? user = auth current user;

// cloud firestore
  Firebase firestore = firebase.instance;
```

Authentication & database operations :-

```
// Authentication
    Future <void> signIn() async {
    await auth.SignIn with Email AND Pass word(
    email = "user@gmail.com", password = "Pass");

// Firestore
    Future <void> add user() {
    return firestore.collection(user).doc(userID);
    set
    ({name: "John Doe", age = 30});
    }
```

* Benefits of using firebase as a backend :-

Firebase allows development to manage & persise user authentication states, offering seamless user experience.

Firebase analytics provides insights into user behaviour and crashlytics reporting crash for the better app stability & performance monitoring.

It integrates seamlessly with flutter firebase effects a generes tree & providing range of SDK and plugins that simply backend of intra-structure based on demand.

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

→ ① Provides secure user authentication using various methods such as email/password, google SignIn & many more.

② A non-SQL real time database that allows for seamless data synchronization access device. It supports complex queries, offline data access & real-time update.

③ An older, Json based database offering real-time synchronization. Its suitable for app requiring a simple Json structure & real-time data updates.

④ Server-less functions that run in response to an events triggered by firebase features, as HTTPs request.

Firestore achieves realtime data synchronization through the use of data libraries when data in firestore database, the associated libraries are notified & UI is automatically updated. This is based on the observer pattern, where the UI components takes the changes to specific data in dataset.

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

→ There are following commonly used widget in flutter and their roles in widget tree :-

ⓐ Container widgets :-

Container widget is a versatile container that can hold & decorate other widgets.

Eg :-

```
dart :
widget build (Build context){
    return  Container (child : Text ("Hello"));
}
```

ⓑ Column & row widgets :-

These widgets organize child widgets vertically (column) or horizontally (row).

Eg :-

```
dart :
widget build (Build Context){
    return  Column (Children : [
                Text ('item 1'), Hello how are you?
                Text ('item 2'), ],);
}
```

ⓒ List-view widgets :-

creates a scrollable list of widgets.

Eg :-

```
dart :
widget build (Build context){
    return  Listview (
            children : [
            List file (title : Text ('item 1')),
```

```
        ListTile (title : Text ('item 2')), ], );
      ) }
```

(2) AppBar Widgets :-

Represents the app bar at the top of the scre...

Eg :-

```
dart :
widget build (Build context) {
  return Scaffold (
          appBar : AppBar (
          Title : Text ('MyApp'),),
          body : );
      }
```