Yogeshchandra Paranjpe
D15B - 52
MPL - A1

Q1)

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

→ The key advantages of using flutter include customizable widget, cross platform capabilities, a fast development cycle and strong community support, because of which it is a popular choice for app development.

(i) single codebase:- Develop for ios & Android from a unified codebase, reducing development time & effort.

(ii) Hot Reload:- Real time code changes without restarting, enhancing development efficiency.

(iii) Rich widget Library:- Pre-designed customizable widget for consistent and visually appealing user interface.

(iv) High Performance:- Flutter compiles to native ARM code and uses the skia graphics engine, ensuring smooth performance.

(v) consistent UI Across platforms:- Flutter ensures a consistent UI by adapting widgets to the design principle of each platform.

Q.1

b) Discuss why flutter framework differs from traditional approaches and why it has gained popularity in the developed community.

→

(i) single codebase :- Flutter uses a single codebase for ios and Android, unlike traditional approaches that require seperate codebases.

(ii) widget based UI :- Flutter utilizes a widget based UI system for consistent design across platforms, while trad. approaches rely on platform based comp.

(iii) Hot Reload :- Flutter's hot reload allows real-time code changes, speeding up development, iterations unlike traditional manual compilation processes.

(iv) Dart Language :- Flutter employs Dart, a language specific for framework, different from platform specific languages used in traditional approaches.

⇒ Reasons for popularity :-

(i) Efficiency & Time savings :- Enables code reuse for multiple platforms.

(ii) consistent UI across platforms :- Ensures uniform UI on IOS & Android.

(iii) Rich widget Library :- This simplifies UI development

(iv) Strong community support :- fosters a growing ecosystem.

(v) cost effectiveness :- Streamlined development process.

Q.2

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

→ • In Flutter, the widget tree is a hierarchial representation of UI components, where each node corresponds to a widget defining a structure & appearance of UI. Widgets serve as the fundamental building block ranging from basic elements like buttons & text to a more complex structure.

• Widget composition is a core concept in flutter, allowing developers to build intricate UI through the assembly of simple & reusable widgets. This process involves combining, nesting & configuring widgets to create a modular components. Developers start with foundational widgets and progressively compose them into more sophisticated structures. The hierarchial arrangement of widgets in the tree mimors the layout & composition of UI.

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

→ commonly used widgets in flutter and their roles in widget tree :-

<1> container widget :-

Role → A versatile container that can hold & decorate other widgets.

Eg:-

```
dart
widget build (BuildContext context) {
    return Container (
        child: Text ('Hello, Flutter!'),
    );
}
```

2) Column & Row widget :- organize child widgets vertically (columns) and horizontally (row).

Eg:-

```
dart
widget build (BuildContainer context) {
    return Column (
        children: [
            Text ('Item 1')
            Text ('Item 2'),
        ],
    );
}
```

3) Listview widget : creates a scrollable list of widgets

Eg:-

```
widget build (BuildContext context) {
return Listview (
children : [
    ListTile (title : Text ('Item 1')]
    ListTile (title: Text ('Item 2')]
    ],
);
}
```

4) Appl^{n} Bar widget:- Represents application bar at the top of the screen.

Eg:-

```
widget build (Build context context){
  return support (
  appBar: AppBar (
    title: Text ('my App')
  ),
  body:
);
}
```

5) Text field widget:- Allows user i/p for text

Eg:-

```
widget build (Build context context){
  return Textfield (
    decoration: Input decoration (
    labelText: 'Enter your name',
  ),
);
}
```

Q.3a) Discuss the importance of ~~flutter in~~ state management in flutter appl^{n}.

→ (i) Dynamic UI: state management is critical for handling dynamic changes in UI, whether its updating UI elements in response to user interactions or reflecting changes in data, effective state management ensures UI remains responsive & reflects the current appl^{n} state.

(ii) code Reusability:- Well managed state enables the creation of modular and reusable components. In flutter where widgets can be composed or reused, effective state management ensures that these components can be easily integrated into diff. parts of appln. promoting a DRY (Don't Repeat Yourself database).

(iii) cross screen communication:- state management facilitates communication betn diff screens or components of an applⁿ, allowing them to share & synchronize data.

(iv) Efficient memory usage:- Effective state management helps optimize memory usage by ensuring that only the necessary components are rebuilt when state changes occur, preventing unnecessary widget rebuilds.

Q.3

b) compare & contrast diff state management approaches available in flutter. Provide scenario where each approach is suitable.

→ .setstate:-

pros:- • simple & built in
• suitable for small apps and quick prototyping

cons:- limited to widgets subtree
can lead to code duplication

suitability:- set state is suitable for small to medium sized.

• Provider:- The provider package is a popular state management solution in flutter.

Pros:- Easy to use and set up

Cons:- might require additional packages.

• Riverpod:- Riverpod is an extension of the 'provider' package with additional features.

Pros:- Based on a simplified syntax & design, making it easy to understand & use.

Cons:- Requires learning a new API

* Scenario:-

⇒ Use 'setstate' when:-

• Dealing with simple state changes within a single widget.

• The appln is small and the overhead of other state management soln is unnecessary.

⇒ Use 'provider' when:-

• Building medium sized applns.

• Dependency injection is a key requirement.

⇒ Use 'Riverpod' when:-

• Building large & complex applications with many widgets & screens.

• A more modular and scalable approach is desired.

Q.4a) Explain the process of integrating firebase with a flutter appl^{n}.

→ • Integrating firebase with a flutter app involves creating a firebase project, registering the app, adding dependencies, and initializing firebase.
• key benefits include a real time NoSQL database (firestore) easy authentication, cloud functions & storage.
• Firebase ensures scalability, handles security and provides analytics tool.

Q.4b) Highlight the firebase services commonly used in flutter development and provide a brief overview of how data synchronization is achieved.

→ In flutter, development essential firebase services include firestore, firebase authentication and firebase cloud functions. firestore, a NoSQL database organizes data into collections and documents.
Firebase authentication ensures user logins with various methods and its realtime synchronization enables dynamic responses to user authentication events like logins or logouts.