YASH USKELWAR
D15B-69.

MAD- Assignment- I.

Q1 a] Explain the key features & advantage of using flutter for mobile app development.

→ Flutter is a cross platform UI toolkit developed by google for building natively compiled application for mobile web & desktop from a single codebase features and advantages include:

- Hot reload, which enables to instantly view changes without restarting the app.
- → Widget based architecture, making the development modular and customisable.
- → Expressive UI; providing rich sets of UI for creating great interface.
- → Single codebase, so develop once, deploy everything reducing the overall time.

b] Discuss how the flutter framework differs from traditional approaches and why it has gained popularity.

→ Flutter uses a reactive framework, whereas tradition approaches are typically imperative.

→ Flutter offers a consistent UI across platform ensuring a native look & feel.

→ The use of Dart lang. & the widgets based approach enhances developers productivity.

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

→ In flutter, the widget is a fundamental concept that represents the hirarchy of user interface elements in an app. Everything in flutter is a widget wheather its a button, text or image. Widget are arranged in tree structure, where each widget can have zero or more children forming a hirarchy.

→ The widget tree is composed of various types of widget, each serving a specific purpose. Widget in flutter can be broadly classified into two :

↳ Statefull widget
↳ Stateless widget

→ Stateless widget are immutable & don't have any internal state, while stateful widget can change their internal state during their lifetime.

Date
Page

(b) Provide examples of commonly used widget & their roles in creating a widget tree.

→ Examples of commonly used widget.

① Material app : Define the basic structure of a flutter app.

② Scaffold : Represents the basic visual structure of the app bar and body.

③ Container : A box model than can contain other widget, providing layout & styling.

④ Row & column : Arrange child widget horizontally or vertically.

⑤ List views : Display a scrolling list of widgets.

⑥ Floating action buttons : Represents of floating action button.

Q3. Discuss the importance of state management of flutter app.

→ State management is a crucial aspect of building robust & effective flutter application. In flutter state reffers to the data that influences the

Teacher's Sign

apperance & behaviour of widget. Making state effectively is essential for creating responsive & dynamic application. Here are some reason.

↳ User interface updates.
↳ Performance optimisation.
↳ Code maintainabitity
↳ Reusability & modularity.
↳ Persistant & navigation
↳ Stateful widget limitation
↳ Concurrency & asynchronous operations

(b) Compare the different state management approaches available in flutter.

→ set state :
Pros.
→ simplicity : 'set state' is the most straighforward way to manage state in flutter. It is built into the frame work & is easy to understand for beginers

→ Apropiate for simple UIs. For smalls to moderately complex UIs where the state changes are localised and the widget tree is not deeply nested 'set state can be sufficient.

⑤ Provide examples of commonly used widget & their roles in creating a widget tree.

→ Examples of commanly used widget.

① Material app : Define the basic structure of a flutter app.

② Scaffold : Represents the basic visual structure of the app bar and body.

③ Container : A box model than can contain other widget, providing layout & styling.

④ Row & column : Arrange child widget horizontally or vertically.

⑤ List views : Display a scrolling list of widgets.

⑥ Floating action buttons : Represents of floating action button.

Q.3]. Discuss the importance of state management of flutter app.

→ State management is a crucial aspect of building robust & effective flutter application. In flutter state reffers to the data that influences the

Teacher's Sign

apperance & behaviour of widget. Making state effectively is essential for creating responsive & dynamic application. Here are some reason.

↳ User interface updates.
↳ Performance optimisation.
↳ code maintainability
↳ Reusability & modularity
↳ Persistant & navigation.
↳ Stateful widget limitation
↳ Concurrency & asynchronous operations

(b) Compare the different state management approaches available in flutter.

→ set state:
Pros.
→ simplicity: 'set state' is the most straigh forward way to manage state in flutter. It is built into the frame work & is easy to understand for beginers

→ appropriate for simple UIs. For smalls to moderately complex UIs where the state changes are localised and the widget tree is not deeply nested 'set state can be sufficient.

Q.4] Explain the process of integrating firebase with a flutter app.

→ 1. Create a firebase project.
→ Go to the firebase console & create a new project
→ Follow the setup instructor

2. Add firebase to flutter project.
→ In flutter project, add the firebase SDK dependencies to the 'yaml' file.

3. Initialise firebase.
→ Import the firebase package & initialize firebase in 'main.dart' file

4. Use firebase services in the app.
→ Implement firebase service in your app code.

Benefits using firebase.

↳ Real time database
↳ Authentication
↳ Cloud function
↳ Cloud firestore
↳ Hosting & analytics
↳ Secure & scalable
↳ Easy setup & integration

Teacher's Sign.:

b] Highlight the firebase services commonly used in flutter development & provide a brief overview.

→ Common firebase service
↳ Authentication : Firebase authentication for user sign-in.
↳ Firestore : A No sql database for real-time data sync.
↳ Firebase cloud messaging : Push notification for engaging more user.

Data sync.
↳ listner and streams. Firebase services use listner and stream extensively. Flutter development can use stream-based APIs to listner for change in data wheather its in firebase, the real time database or other firebase service.

↳ Offline support : Firebase services provide built in offline support. Flutter apps can work seamlessly offline and when connectivity is restored, changes made offline are automatically sync with the server.