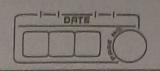
Raghvendra Tripathi DATE !-! D15B - 68 MPL. Assignment 1 Q.1.) a) Explain the key features and advantages of using flutter for mobile app development.

That is a cross platform UI toolkit developed by Google for building natively compiled applications for mobile, web and desktop from a single codebase least features and advantages include: 1. Hot Reload: Enables developers to instantly, view changes without restarting the app.

2. Widget-based Architecture: UI components in Flutter are widgets, making the development modular and customizable. 3. Expresibe: flutter provides a rich set of customizable widgets for creating visually appealing interfaces 4. Single Codebase: Berelop once, deploy everywhere, reducing development time and effort. b.) Discuss how the flutter framework differs from traditio nal approaches and why it has gained popularity in the developer community. 1. Flutter uses a reactive framework , whereas traditional approaches are typically imperative.

2. Flutter offers a consistent UI across platform ensuring a native look and feel. 3. The use of Dart language and the widget based approach enhances developer productivity.

FOR EDUCATIONAL USE

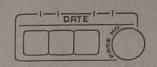


(P.2.) a) Describe the concept of the widget tree in flutter.

Explain how widget composition is used to build complex eyer interfaces. 1) In flutter, the widget is a fundamental concept that represents the hierarchy of user interface elements in an application. Everything in Flutter is a widget whether its a button, text, image or even more children, forming the hiorarchy. 2) The widget free B composed of various types of widget, each serving a specific purpose widgets in Flutter can be broadly categorized sito stateless and stateful -nel state, while stateful widgets can change their interpal state during their exterime. b.) Provide examples of commonly used widgets and their roles in creating a widget true Framples of commonly used widgets.

1. Material App: Defines the basic structure of a no 2. Scatfold: Represents the basic visual structure of the app, including the app bar of body 3. container: A box model than can contain other widget, providing layout & styling.
4. Row & column: Arrange child widgete homeontally or vectically. 5. Gstriew: Displays a scrolling est of widget

FOR EDUCATIONAL USE



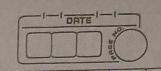
(0.3.) a) Discuss the importance of state management in Flutter applications!

state management: Is a crucial aspect of building robust and efficient flutter applications. In flutter state' refers to the data that influencers the appearance and behaviour of widgets. Managing state effectively is essential for creating responsive, dynamiz and scalable applications. Here are some key reasons why state management is important in flutter.

- 1. Usor Interface Updates
- 2. Performance optimization.
- 3. Code Maintai nability
- 4. Reusali By and Modularity
- 5. Persistance and Navigation
- 6. Stateful widget cimitations
- 7. Concurrency and Asynchrous Operations
- b) compare and contrast the different state management approachs available in flutter such as setstate provider and knowpood. Provide scenarios where each approach is suffable.
- 1. set State!

Brosser

- Simplicity: 'setState' B the most straight forward way to manage state in flutter. It is built into the frame work and is easy to understand for Legineurs. setState is appropriate for simple UI's. For small to moderately complex UI's where the state changes are



localized and the widget tree is not deeply nested 'set Stati' con be sufficient. Suitable Scenariost - small to moderately sized opps. - simple UI's with limited interactivity - Learning and prototyping purposes 2. Provider! - Provider allows for stoped and localized state man - agement, reducing the need for prop drilling.

- It is easy to integrate into flutter app's and
offers a good balance between simplicity and
fluxibility.

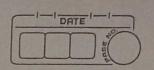
- Provider B widely used and has good community
support: support. sufable scenariost - Apps of varying sizes with moderate to complex UI's - situations where a centralized state management solution is needed but without the complexity of other solutions. 3. Riverpod.

- It is scoped and flexible.

- It is immutable and reactive also provides inheritance. suitable scenarios !-

- Large and complex applications.

- stuations where a more sophisticated scalable I reactive state management solution is required



- Projects where dependency injection is original.

(0.4) a) Explain the process of integrating Firebose with a Flutter application. Discuss the benefits of using firebose as a backend solution.

-> 1. create a Grebase project.

- Go to the Firebase Console and create a new Project - Pollow the setup instructions.
- 2 Add Grebase to Fluttor project.
- In your flutter project, add the Grebase SDK dependencies to the 'yaml' file-3. Instralize firebase!

- Import the firebase packages and artialize forebase in the main-don't file.
- 4. Configure Prebase services +

Depending on the sources you want to use (authentica-- tion, freston, etc), configure them by following the specific solup instructions provided by firebase

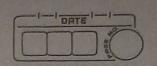
5. Use Frebase service in the App:

- Implement Brebase sources in your app code

Benefits of using firebaser

- 2. AuthentiZation
- 3. Cloud functions
- 4. Cloud fireston
- 5. Frebase storage
- 6. Hosting or and analys 1).





- b) Highlight the firebase sources commonly used in

 Flutter development and provide brief ovorview of
 how dota synchronization is achieved:

 common Arebase services in flutter over

 1. Authenticentron: Firebase authentication for user sign-is
 2. Firestore + A NoSal database for real-time data sync.

 3. Firebase (loud Messaging (pcm): push notifications
 for engaging users.
 - Dota Synchronization
 Firebase services use listeners and streams extensively.

 Flutter developers can use stream-based API's to listen

 for changes in data, whether it's in firestor, the

 Real time Database, or other firebase services.
 - leactively updating UII fluttors 'Stream Buildor' widget is commonly used to reactively update UI components based on the changes in data streams. When data changes on the servor, the stream emils new data triggoring a rebuild of the associated UI.
 - offine Support: finbase services provide built-in office support. Plutter apps can work seamlisty offine of when connectivity is restorced, changes made offline are automatically synchronized with the server.