NAME: KANAK PENDSE
D15B - 54.
MAD-ASSIGNMENT-1.

NAME:_____ STD.:_____ DIV.:_____

DATE :
PAGE :

Assignment-1.

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

→ (i) Single code base: Develop for ios and android from a unifide codebase, reducing development time and efforts.

(ii) Hot reload: Real time code changes without restarting the run process, enhancing development efficiency.

(iii) Rich widget library: Pre-designed, costomizable wegdets for consistant and visually appealling user interface.

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

(v) 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

→ (i) Dart language: Flutter employs dart, a language specific to framework, different from platform specific languages used in traditional approach.

(ii) Efficient and Time saving: Flutter reduces development time for ios & android by enabaling Single code base.
Also reduce development time by implementing code

NAME:_____ STD.:_____ DIV.:_____ DATE: PAGE:

reusability.

(iii) 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.

NAME:_____ STD.:_____ DIV.:_____ DATE : PAGE :

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 in user interfaces. wether its updating UI elements in response to user interactions or reflecting changes in data, effective statuctre management ensures that the UI remains responsive & reflects the carrent applicati-on state.

- Code reusability: well-monged 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, promoting a DRY codebase
- Cross screen communication: state management faciliate 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 scenario s where each approach is suitable.

→ ① set state: The setstate method is a build-in mechanism in flutter for manging the internal

NAME:_____ STD.:_____ DIV.:_____ DATE: PAGE:

State of stateful widget:
Scenarios: setstate is suitable for small to moderatly complex UI's where state changes are localized to specific widget & don't to be shared across the entire application.

(ii) Provider: The provider package is a popular & light weign state management solution in flutter. It follows the provider pattern & is base is on inherited widget.
Scenarios: Provider is suitable for managing state within specific parts of widget tree, creating a scoped and efficient solution.

(iii) 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 application: 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.

→ (1) Create firebase & Project:
Start by creating a project on fire base console & configure your app. Add fire base to flutter project:

NAME:_____ STD.:_____ DIV.:_____ DATE: PAGE:

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(BuildContext context){
  return Container(
   Child: Text('Hello, world'),
  );
}
```

ⓑ Column & row widgets:
These widgets organize child widgets vertically (Column) or horizontaly (row)

eg,
dart

```
widget build(BuildContext context){
  return Column(
   Children:[
    Text('Item1'), Hello How are you.
    Text('Item2'),
   ],
  );
}
```

NAME: STD.: DIV.: DATE : PAGE :

© Listview widgets :
creates a scrollable list of widgets.
Eg,
dart
```
Widget build( BuildContext context) {
  return ListView(
    children:[
     ListTile( title: Text ('Item1'),
     ListTile( title: Text('Item2'),
  ],
 );
}
```

AppBar Widgets :
Represents the app bar at the top of the screen
Eg,
dart
```
widget build( BuildContext context) {
  return Scaffold(
   AppBar : AppBar(
   title : Text ('My App'),
  ),
   body :
 );
}
```

NAME:_____ STD.:_____ DIV.:_____ DATE : PAGE :

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

yaml :

```
dependencies :
  firebase_core : ^ letest_version
  Firebase_auth : ^ Letest_ version
  Cloud_ fireston : ^ Letest version
```

Run flutter pub get to fetch the dependencies.

Initialize firebase in your flutter app by calling firebase.initializeApp()
in the main() method :-

dart :-

```
Import package: 'Firebas_core/firebase_core.dart';

Void main() async{
widgets flutter Binding. ensure initialized();
await firebase. initialize App();
 runApp (MyApp());
};
```

Use firebase service like authentication, firestore in flutter app by importing the relevant packages and initializing them using Firebase project credentials dart:

NAME: STD.: DIV.: DATE: PAGE:

```
import 'package:firebase_auth/firebase_auth.dart';
import 'package:cloud_firestone/cloud_firestone.dart';

// Authentication
FirebaseAuth auth = FirebaseAuth.instance;
User? user = auth.currentUser;

// Cloud firestone
Firebase firestone firestone = Firebase.instance;
```

Authentication & database operations:

```
// authentication
Future<void> signIn() async {
 await auth.SignInWithEmailAndPassword(
 email:'user@example.com, password:"Pass");

// Firestore
   future<void> addUser() {
 return firestore.collection('user').doc(userID')
.set
({'name': 'John Doe', age: 30});
}
```

Benifits of using firebase as a backend:

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

NAME:_____ STD.:_____ DIV.:_____ DATE : PAGE :

- Firebase analytics provides insights into user behaviour and crashlytics reporting crash for the better app stability and performance monitoring.
- It integrates seamlessly with flutter, fire base effects a generes tree tire & providing ~~str~~ an range of SDK, and plugins that simply backend of intrastruture based on demand.

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

→ (i) Provides secure user authentication using various methods such as email/password, google sign-in & many more.
(ii) A non-SQL, real time database that allows for seamless data synchronization access device. It supports complex queries, offline data access & real-time update.
(iii) An older, JSON-based database offering real-time synchoonization. Its suitable for application requiring a simple JSON structure & real time data updates.
(iv) Server-less functions that run in response to an events triggred by fire base feature

NAME:_____ STD.:_____ DIV.:_____ DATE : PAGE :

as HTTPs request.

• Firestone achieves real time data synchronization through the use of data libraries. when data in Firestone 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.