Name : Anya Manoj Madhavi
Class : D15B
Roll no : 29

05 AH

MPL Assignment 1

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

→ Key features of Flutter :
① Single Codebase - Write one code for both Android and ios
② Fast Performance - Uses Dart Language and high performance rendering engine.
③ Hot Reload - See changes instantly without restarting the app.
④ Rich UI Components - Comes with customizable widgets for smooth UI design.
⑤ Native like experience - Provides high quality animations and fast execution.
⑥ Cross - Platform Support - Can be used for mobile, web and desktop apps.
⑦ Open - Source - Free to use and has a strong developer community.

Advantages of using Flutter :
① Saves Time and Effort - Single codebase for multiple platforms.
② Cost effective - Reduces development cost and time.
③ Attractive UI - Provides beautiful and customizable widgets.
④ Good Performance - Uses Dart and Skia for fast and smooth rendering.
⑤ Easy Integration - Supports third party plugins and native code integration.
⑥ High Speed Development - Hot reload features speeds up coding.

Q1. b) Discuss how the Flutter framework differs from traditional approaches and why it has gained popularity in the developer community

→ How Flutter differs from Traditional Approaches:

① Single Codebase - Traditional methods need separate code for Android (Java / Kotlin) and iOS (Swift / Objective-C), but Flutter uses one code for both.

② Hot Reload - Traditional apps require full restart after changes, but Flutter updates instantly.

③ UI rendering - Flutter has its own rendering engine (Skia) for faster performance.

④ Performance - Flutter compiles directly to native machine code, making it faster than frameworks that use a bridge.

⑤ Customization - Traditional UI design depends on platform - specific components but Flutter provides fully customizable widgets.

Why Flutter is popular among developers:

① Fast Development - Hot reload and single codebase save time.

② Cross-Platform Support - works on mobile, web and desktop.

③ Beautiful UI - Rich, customizable widgets for modern designs.

④ High performance - runs smoothly without a bridge like React Native

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

→ Concept of widget tree in Flutter :

① In flutter, everything is a widget, widgets are arranged in tree structure, called the widget tree. This tree represents the UI of app, where parent widgets contains child widgets.

② Eg : A Scaffold widget can have a Column widget, which contains Text and Button widgets. Changes in widgets update the tree dynamically.

③ Widget Composition for Complex UI :
Flutter uses small, reusable widgets to build complex UI. Instead of creating a single large UI block, developers combine multiple small widgets.

④ Eg : 1) A ListView can contain multiple Card widgets
2) A Column can hold Text, Image and Buttons.

This modular approach makes the UI flexiable, readable and easy to manage.

Q.2. b) Provide examples of commonly used widgets and their roles in creating a widget tree

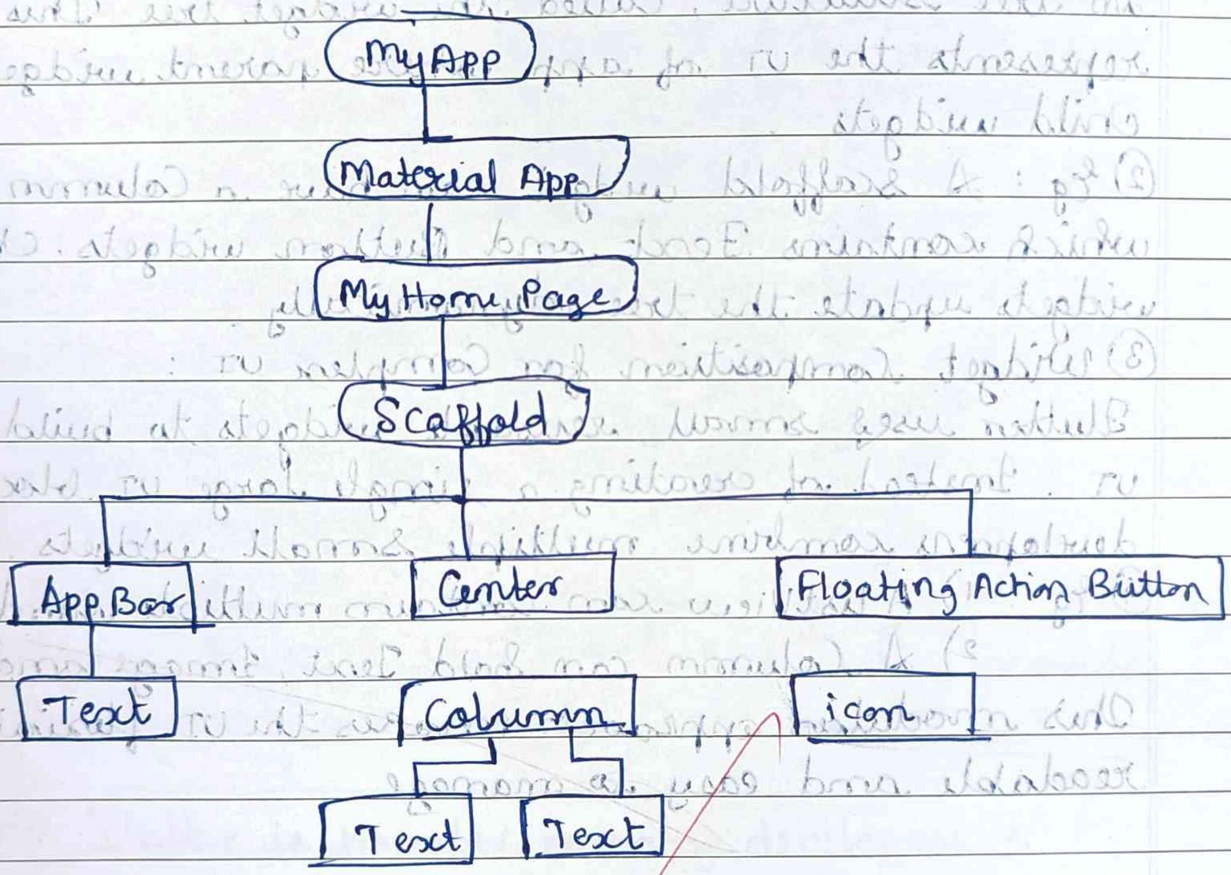
→ Commonly used widgets and their roles in a widget tree :

① Scaffold : Provides the basic layout structure

② App Bar : Displays the top navigation bar with a title.

③ Text : Displays simple text on the screen.
④ Image : Shows images from assets or URL's.
⑤ Container - Used for styling.
⑥ ListView - Displays scrollable lists.

Example widget Tree :



This tree structure helps in organizing and managing the UI efficiently.

Q.3.a) Discuss the importance of state management in Flutter applications.

→ Importance of State Management in Flutter Applications.
~~Staten~~ State management is important because it controls how the app stores, updates and displays data when the user interacts with it.

Why State Management is Needed?
① Keeps UI updated
② Improves performance
③ Manages Complex Data
④ Ensures Smooth User experience.

Types of State in Flutter

① Local State - Managed within a single widget using Stateful widget.

② Global State - Shared across multiple screens using Provider, Riverpod, Bloc or Redux.

Without proper state management, the app may behave unpredictably or show outdated data.

Q.3.b) Compare and contrast the difference state management approaches available in Flutter, such as setState, Provider and Riverpod. Provide scenarios where each approach is suitable

Approach	How it works	When to use
setState	Updates UI by calling setState() in a Statefullwidget	Best for small apps or managing state within a single widget. eg: Toggling a button color.
Provider	Uses Inherited widget to share state across widgets efficiently.	Suitable for medium sized apps where data needs to be shared between multiple widgets. eg: Managing user authentication.
Riverpod	An improved version of Provider with better performance and simpler syntax	Best for large apps that need complex state management with dependency injection. eg: Handling API data and app wide themes.

Q.4. a) Explain the process of integrating Firebase with a Flutter application. Discuss the benefits of using Firebase as a backend solution.

→ Process of Integrating firebase with Flutter Application :

① Create a Firebase Project - Go to Firebase Console, create a new project.
② Add Firebase to Flutter App. - Register the app (Android/ ios) and download the google services json (Android) or Google Service - Info. plist (ios).
③ Install Firebase Packages - Add dependencies like 'firebase-core' and 'firebase-auth' in 'pubspec.yaml'.
④ Initialize Firebase - Import Firebase in 'main.dart' and call 'Firebase. initializeApp()'.
⑤ Use Firebase Services - Implement authentication, database or cloud functions as needed.

Benefits of using firebase as a Backend Solution :
① Real-time Database
② Authentication
③ Cloud Firestore
④ Hosting and Storage
⑤ Scalability
⑥ Push Notifications.

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

→ Common Firebase Services used in Flutter Development.

① Firebase Authentication
② Cloud Firestore
③ Firebase Realtime Database
④ Firebase Cloud Storage
⑤ Firebase Cloud Messaging (FCM)
⑥ Firebase Hosting
⑦ Firebase Analytics

How data Synchronization is Achieved

① Real time updates - Firestore and Realtime Database sync data across devices instantly.
② Listeners & Streams - widgets listen for changing and updates the UI automatically.
③ Offline Support - Firebase caches data, allowing apps to work offline and sync when online.

This ensures fast, smooth and automatic data updates in Flutter apps.