Page No. 1
Class D15B
Roll No. 09
Date: / /201

Vaishnavi Chaudhary
Roll no. 09
D15B
Assignment no. 1 (MAD)

Q.1 a) Explain key feautures and adv of using Flutter for mob. app dev.

• Features of Flutter:

1 Fast development cycle - allows developers to see changes to app in real-time as they make modifications to code.
2. customizable widgets - to create beautiful and user-friendly interfaces.
3. Performance - offers fast and smooth animations and transitions and ability to run smoothly on older devices.
4. Flexibility and control - design and development process, making it attractive

• Advantages of Flutter:

1. Fast development :- Fluter's cycle allows developer to see changes to app in real-time as they make modifications to code. This can greatly increase speed and efficiency of development process of apps.
2. Beautiful User Interfaces - It provides a rich set of customizable widgets that can be used to create beautiful and user-friendly

Page No.	
Class	
Roll No.	
Date:	/ /201

interfaces Framework also offers strong emphasis on design and visual appeal, making it an attractive choice.

3. High performance - Flutter offers fast and smooth animations & transitions, & is designed to run smoothly on older devices. The framework is optimized for performance, making it an attractive choice. As a result no. of targeted users increases.

4. Cross - Platform Development - Flutter supports not only mob. app dev. but also web & desktop app dev. This makes it versatile tool for developing apps that need to run on multiple platforms without any issues.

5. Open - Source - Flutter is free & open - source framework, making it accessible to wide range of developers & companies.

Q.1. b) Discuss how Flutter framework differs from traditional approaches & why it has gained popularity in developer community.

Flutter is gaining popularity rapidly. This is because Flutter offers no. of adv. over other mob. dev. frameworks such as cross - platform dev, native performance and hot reload.

1. Single Codebase for multiple platform - Unlike traditional approaches where separate codebases are req. for iOS & Android, Flutter allows developers to write code once & deploy it across multiple platforms. This significantly reduces dev. time, effort & maintenance overhead.
2. UI Rendering - Traditional approaches often use native UI components specific to each platform, resulting in differences in appearance & behavior between iOS & Android apps. Flutter uses its own rendering engine to create custom UI components that look & feel native on each platform.
3. Hot Reload - Flutter's hot reload feature enables developers to instantly see effects of code changes without restarting app. This accelerates dev. process by allowing developers to iterate quickly, experiment with diff designs, & fix bugs on the fly.
4. Access to Native Features :- While Flutter provides a rich set of customizable widgets, it also offers plugins that allow developers to access native device features & APIs. This enables integration with device functionalities like camera, GPS, sensors & more providing seamless user experience.

Page No.	
Class	
Roll No.	
Date: / /201	

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

A widget is a key component of UI in Flutter. Widgets are used to describe structure & layout of your app's UI elements, such as buttons, text fields, images, containers & more.
Widget Tree - Flutter apps are constructed using hierarchy of widgets known as widget tree. At root of tree is Material App or CupertinoApp, which defines overall structure of app. Within this three, widgets nest inside one another to create complex UIs.
Widget Composition - We can compose complex UIs by combining multiple widgets. For eg, we can nest widgets inside columns, rows, containers, & more to create layouts. Flutter provides a rich set of layout widgets to help you structure your UI.

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

1 Container - A versatile widget used for layout & styling. It can contain other widgets & apply properties like padding, margin & decoration.

2. Row - Arranges its children widgets horizontally in a row.
3. Column = Arranges its children widgets vertically in a column.
4. Stack - Allows widgets to be stacked on top of each other. Useful for creating overlapping UI elements.
5. ListView - Displays a scrollable list of children widgets. Ideal for displaying a large no. of items efficienty.
6. AppBar - Represents app bar at top of screen Typically contains actions, titles, & navigation controls.
7. Text : Displays text on the screen with customizable styles.
8. Image - Displays images from various source such as assets, network or memory.
9. Button - Represents interactive buttons that users can tap on, Examples Include FlatButton, RaisedButton, & IconButton.
10. TextField - Allows users to input text. Provides opt for customizing i/p constraints, validation & keyboard type.
11. Checkbox - Represents checkbox that users can toggle on or off.
12. RadioButton - Displays group of radio buttons allowing users to select one opt. from multiple choices.

Page No.	
Class	
Roll No.	
Date:	/ /201

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

Imagine building an app without any mechanism to handle the state. Every time something in your app changes, you need to manually update the UI to reflect those changes. This approach quickly becomes chaotic, error-prone, & unsustainable as your app grows in complexity.
This is where state management comes into play. It's the practice of efficiently managing & updating the state of your application. State management solutions provide structured ways to handle state changes & ensure that your app remains responsive & consistent.

Q.3 b) Compare & contrast diff state management approaches available in Flutter, such as setState, Provider & Riverpod. Provide scenarious where each approach is suitable

1. setState -
Description: 'setState' is a built-in method provided by Flutter for managing state of a widget. It allows you to update the state of widget & trigger a rebuild of widget & its children.

Suitable Scenarios :
- For simple apps with minimal state management needs.
- When the state to be managed is local to a single widget & does not need to be shared with other widgets.
- For small-scale projects or when starting with Flutter to quickly prototype or experiment

2. Provider -
Description : Provider is popular state management solution in Flutter that offers a simple & efficient way to manage application-wide state & share it across diff parts of the app.
Suitable Scenarios :
- For medium to large-scale applications where state needs to be shared across multiple widgets.
- when there's a need for scoped or global state management.
- When you want to avoid prop drilling.

3. Riverpod -
Description : Riverpod is an advanced state management library for Flutter, built on top of Provider.

Page No.
Class
Roll No.
Date: / /201

Suitable Scenarios:
- For large-scale applications with complex state management requirements.
- When you need fine-grained control over the lifecycle of state objects & their dependencies.
- When working on projects that require testability & maintainability
- For managing complex asynchronous workflows & data dependencies.

Comparison:
1 Complexity - 'setState' is simplest approach, while Riverpod tends to be more complex due to its additional feautures and concepts like dependency injection.
2. Performance - Riverpod typically offers better performance optimizations compared to 'setState' & 'Provider', especially for large-scale applications
3. Flexibility - Provider & Riverpod offer more flexibility in managing state compared to 'setState', especially when dealing with complex manipulation & data flow.
4. Community Support - Provider has a larger community & ecosystem.

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

1. Create Firebase project - Go to Firebase console create new project & register your Flutter app with Firebase by providing package.
2. Add Firebase ~~SDK~~ - Add Firebase SDK dependencies to your Flutter project by including necessary packages in your 'pubspec.yaml' file
3. Configure Firebase Services - Configure Firebase services you want to use, Firebase Authentication for user authentication, Firebase Cloud Messaging. Set up necessary configurations & rules.
4. Initialize Firebase - Initialize Firebase in your Flutter app by calling 'Firebase.initializeApp()' in your main function.
5. Use Firebase Services - One Firebase is initialized, you can start using Firebase services in your Flutter app.

Page No.
Class
Roll No.
Date: / /201

• Benefits of using Firebase as backend:

1. Real time Database - Firebase provides real-time database solutions like Firestore, which allow seamless data synchronization
2. Authentication - Firebase Authentication offers easy-to-use authentication meth including making it simple to implement user authentication in Flutter apps.
3. Cloud Functions - Firebase allows you to extend your App's functionality with Cloud Functions, enabling you to run backend code.
4. Scalability - Firebase is scalable platform that can handle large no. of users & data, making it suitable for enterprise.
5. Offline Support - Firebase offers offline support for Firestore & Real time Database.

Q.4. b) Highlight Firebase services commonly used in Flutter dev. & provide a brief overview of how data Synchronization is achieved.

1. Firestore:
   • Firestore is flexible, scalable database for mobile, web. & server dev. It

allows you to store & synchronize data in real-time between your Flutter app & Firebase Cloud.
- It uses real-time synchronization mechanism based on WebSockets. When data changes on the server, Firestore sends real-time updates to all connected clients, including your Flutter app.

2. Firebase Authentication:
- Firebase Authentication provides secure & easy to use authentication methods.
- Authentication data is managed securely by Firebase on its servers. When a user signs in or signs out from your Flutter app, Firebase Authentication handles authentication process transparently.

3. Firebase Cloud Messaging (FCM):
- FCM allows you to send push notifications to your Flutter app on Android, iOS, & web devices.
- Data Synchronization - When a push notification is sent from Firebase Console or your server, Firebase Cloud Messaging ensures that notification is delivered to intended devices.

Page No.	
Class	
Roll No.	
Date:	/ /201

4. Firebase storage -
   - Firebase storage provides secure & scalable storage solutions for user-generated content such as images, videos & files
   - When files are uploaded to Firebase Storage from your Flutter app, they are stored securely in cloud. Your app can then retrieve & download these files as needed.