SVKM'S

Mithibai College of Arts, Chauhan Institute of Science &
Amrutben Jivanlal College of Commerce and Economics (Autonomous)
Academic Year (2022-23)

Class: S. Y. B.Sc.(Computer Science) **Semester: IV**

Program: Bachelor of Science **Max. Marks: 75**

Course Name: Android Developer Fundamentals **Time: 10:30 am to 1:00 pm**

Course Code: USMACS404 **Duration: 2hrs 30 minutes**

Date:

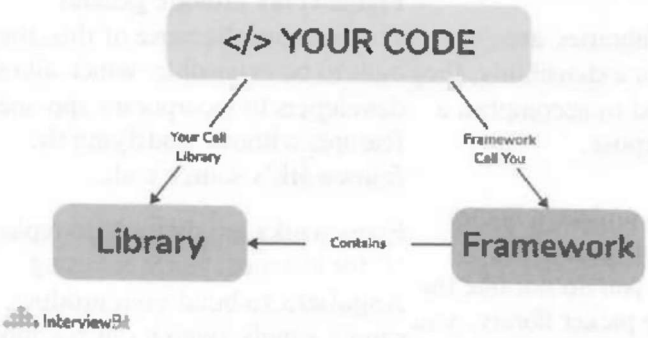
MODEL ANSWER KEY SET 3

Q.1. Attempt any Three **(21)**

a. With the example differentiate framework and library (7)

By using a library, you control the flow of the program. The library can be invoked whenever and wherever you like. Contrary to this, when you use a framework, the flow is controlled by the framework. The framework instructs you where to put your code, but it will call your code as required. Simply put, our code calls the library's code, but in a framework, it's the framework's code that calls our code as shown in the below diagram.

Developers can invoke libraries to perform specific tasks by using components, classes, and methods. A framework, however, already provides code to perform common tasks and uses code provided by a developer for custom functionality.



Parameters	Library	Framework
Definition	Libraries provide developers with predefined functions and classes to make their work easier and boost the development process.	Framework, on the other hand, is like the foundation upon which developers build applications for specific platforms.
Inversion of Control	By using a library, you can control the flow of the application and call the library.	In contrast, when you use a framework, the control is inverted, i.e., the framework controls the flow and calls your code.
Collection	Generally, libraries are a collection of helper modules, objects, classes, functions, message templates, pre-written code, etc.	Frameworks consist of a lot of APIs, compilers, toolsets, support programs, libraries, etc.

Code Modification	Codes in libraries are geared toward a particular program or to solve a specific development problem. Therefore, developers must modify library code to meet their needs.	Despite the fact that frameworks generate new codes for developers. These codes cannot be altered or modified later. Unlike libraries, frameworks do not allow users to modify their pre-written codes, so you don't have to worry about deleting or changing them.
Scope	It is possible to call a library out of context. You may use the library wherever you see fit in your code.	On the other hand, you can only call and use what belongs to a Framework within the same Framework.
Function	In the program linking and binding process, they play an important role.	Using them, you can build and deploy applications in a standard way as the framework already provides code to perform common tasks and uses code provided by a developer for custom functionality.
Complexity	Having a library means understanding the functionality of each method, and it isn't easy to create complex interactions since you need to call many methods to get the desired results.	Frameworks, on the other hand, embody the basic flow, and since plugins need to be added to code, it is easier to do the right modification.
Extensibility	Generally, libraries aren't designed for extensibility; they are designed to accomplish a specific purpose.	Frameworks provide general functionality. Because of this, they are built to be extensible, which allows developers to incorporate app-specific features without modifying the framework's source code.
Replaceable	It is easy to replace a library with another library. For instance, if you do not like the jQuery date picker library, you can use another date picker like a bootstrap date picker or pick date.	Frameworks are difficult to replace. If, for instance, you were using AngularJS to build your product, you cannot simply swap it out for another framework. It requires rewriting the entire codebase.
Performance	Less code is required to build libraries, which leads to faster loading times and better performance.	Developing a framework requires a lot of coding, which increases loading times and decreases performance.
Usage	The purpose of libraries is to perform a defined and specific task. Eg: Image manipulation, network protocols, math operations, etc.	Frameworks can be used for performing a wide range of tasks. Among these are Web application systems, plug-in managers, GUI systems, and so on.
Existing Projects	You can integrate libraries seamlessly into existing projects to add functionality.	Incorporating frameworks seamlessly into an existing project is impossible.

		Instead, frameworks should be used when starting a new project.
Benefits	Good code quality, reusability, and control, enhanced speed and performance of the program, etc.	Faster programming, support from the community, great support for MVC (Model View Controller) pattern, etc.
Examples	JQuery, React JS, etc.	Spring, NodeJS, AngularJS, Vue JS, etc.

Explanation and difference 7 marks

b. Explain Scroll View and write a code snipped for the same (7)

n Android, a ScrollView is a view group that is used to make vertically scrollable views. A scroll view contains a single direct child only. In order to place multiple views in the scroll view, one needs to make a view group(like LinearLayout) as a direct child and then we can define many views inside it. A ScrollView supports Vertical scrolling only, so in order to create a horizontally scrollable view, HorizontalScrollView is used. Android using Kotlin.

Step 1: Create a new project

Click on File, then New => New Project.

Choose "Empty Activity" for the project template.

Select language as Kotlin.

Select the minimum SDK as per your need.

Step 2: Modify strings.xml

Add some strings inside the strings.xml file to display those strings in the app.

strings.xml

```
<resources>
   <string name="app_name">gfgapp_scrollview</string>
   <string name="scrolltext">Kotlin is a statically typed,

            Explanation:
            Here, we have initialized the constructor
            parameters with some
            default values emp_id = 100 and emp_name = "abc".
            When the object emp is created we passed the values for
            both the parameters so it prints those values.
            But, at the time of object emp2 creation,
            we have not passed
            the emp_name so initializer block uses
            the default values and
            print to the standard output.</string>
</resources>
```

Step 3: Modify activity_main.xml

Add the ScrollView and inside the ScrollView add a TextView to display the strings that are taken in the strings.xml file.

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  tools:context=".MainActivity">

  <ScrollView
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:layout_editor_absoluteX="0dp"
    tools:layout_editor_absoluteY="-127dp">

    <TextView
      android:id="@+id/scrolltext"
      style="@style/AppTheme"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:text="@string/scrolltext"
      android:textColor="@color/green"/>
  </ScrollView>
</androidx.constraintlayout.widget.ConstraintLayout>
```

Steps with code 7 marks

c. Differentiate List View and Recycle View (7)

In Android View is a basic building block of UI (User Interface). A view is a small rectangular box that responds to user inputs. RecyclerView and ListView are the two major Views in Android. So in this article, we are going to see the major differences between these two views.

RecyclerView

RecyclerView is a ViewGroup added to the android studio as a successor of the GridView and ListView. It is an improvement on both of them and can be found in the latest v-7 support packages. It has been created to make possible the construction of any lists with XML layouts as an item that can be customized vastly while improving on the efficiency of ListViews and GridViews. This improvement is achieved by recycling the views which are out of the visibility of the user. For example, if a user scrolled down to a position where items 4 and 5 are visible; items 1, 2, and 3 would be cleared from the memory to reduce memory consumption.

To read more on RecyclerView refer to this article: RecyclerView in Android with Example

ListView

A ListView is a type of AdapterView that displays a vertical list of scroll-able views and each view is placed one below the other. Using the Adapter, items are inserted into the list from an array or database. For displaying the items in the list method setAdaptor() is used. setAdaptor() method conjoins an adapter with the list. Android ListView is a ViewGroup that is used to display the list of items in multiple rows and contains an adapter that automatically inserts the items into the list.

RecyclerView	ListView
The RecyclerView's adaptor forces us to use the ViewHolder pattern. The views are split into onCreateViewholder() and onBindViewholder() methods.	The ListView doesn't give that kind of protection by default, so without implementing the ViewHolder pattern inside the getView().
Efficient Scrolling, we can choose the way of scroll-like vertically or horizontally and grids.	Inefficient scrolling, we can only create vertical scrolling.
Use of less memory.	More memory is used for a long list. Sometimes devices get hanged.
Animations using ItemAnimator are easy and smooth. Animations like list appearance and disappearance, adding or removing particular views, and so on.	It's complex to use Animation and hard to handle it.
Dividers between items are not shown by default.	Dividers between items are shown by default.
Use ItemDecorations to add margins and draw on or under an item View.	ItemDecorations require customization.
Explanation 2 marks difference 5 marks	

d. Write a code to add toggle button and display the toggle value (7)

```
import 'package:flutter/material.dart';

const List<Widget> fruits = <Widget>[
  Text('Apple'),
  Text('Banana'),
  Text('Orange')
];

const List<Widget> vegetables = <Widget>[
  Text('Tomatoes'),
  Text('Potatoes'),
  Text('Carrots')
];

const List<Widget> icons = <Widget>[
```

```
  Icon(Icons.sunny),
  Icon(Icons.cloud),
  Icon(Icons.ac_unit),
];

void main() => runApp(const MyApp());

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  static const String _title = 'ToggleButtons Sample';

  @override
  Widget build(BuildContext context) {
    return const MaterialApp(
      title: _title,
      home: ToggleButtonsSample(title: _title),
    );
  }
}

class ToggleButtonsSample extends StatefulWidget {
  const ToggleButtonsSample({super.key, required this.title});

  final String title;

  @override
  State<ToggleButtonsSample> createState() => _ToggleButtonsSampleState();
}

class _ToggleButtonsSampleState extends State<ToggleButtonsSample> {
  final List<bool> _selectedFruits = <bool>[true, false, false];
  final List<bool> _selectedVegetables = <bool>[false, true, false];
  final List<bool> _selectedWeather = <bool>[false, false, true];
  bool vertical = false;

  @override
  Widget build(BuildContext context) {
    final ThemeData theme = Theme.of(context);

    return Scaffold(
      appBar: AppBar(title: Text(widget.title)),
      body: Center(
        child: SingleChildScrollView(
          child: Column(
```

```
mainAxisSize: MainAxisSize.min,
mainAxisAlignment: MainAxisAlignment.center,
children: <Widget>[
 // ToggleButtons with a single selection.
 Text('Single-select', style: theme.textTheme.titleSmall),
 const SizedBox(height: 5),
 ToggleButtons(
  direction: vertical ? Axis.vertical : Axis.horizontal,
  onPressed: (int index) {
   setState(() {
    // The button that is tapped is set to true, and the others to false.
    for (int i = 0; i < _selectedFruits.length; i++) {
     _selectedFruits[i] = i == index;
    }
   });
  },
  borderRadius: const BorderRadius.all(Radius.circular(8)),
  selectedBorderColor: Colors.red[700],
  selectedColor: Colors.white,
  fillColor: Colors.red[200],
  color: Colors.red[400],
  constraints: const BoxConstraints(
   minHeight: 40.0,
   minWidth: 80.0,
  ),
  isSelected: _selectedFruits,
  children: fruits,
 ),
 const SizedBox(height: 20),
 // ToggleButtons with a multiple selection.
 Text('Multi-select', style: theme.textTheme.titleSmall),
 const SizedBox(height: 5),
 ToggleButtons(
  direction: vertical ? Axis.vertical : Axis.horizontal,
  onPressed: (int index) {
   // All buttons are selectable.
   setState(() {
    _selectedVegetables[index] = !_selectedVegetables[index];
   });
  },
  borderRadius: const BorderRadius.all(Radius.circular(8)),
  selectedBorderColor: Colors.green[700],
  selectedColor: Colors.white,
  fillColor: Colors.green[200],
  color: Colors.green[400],
```

```
            constraints: const BoxConstraints(
              minHeight: 40.0,
              minWidth: 80.0,
            ),
            isSelected: _selectedVegetables,
            children: vegetables,
          ),
          const SizedBox(height: 20),
          // ToggleButtons with icons only.
          Text('Icon-only', style: theme.textTheme.titleSmall),
          const SizedBox(height: 5),
          ToggleButtons(
            direction: vertical ? Axis.vertical : Axis.horizontal,
            onPressed: (int index) {
              setState(() {
                // The button that is tapped is set to true, and the others to false.
                for (int i = 0; i < _selectedWeather.length; i++) {
                  _selectedWeather[i] = i == index;
                }
              });
            },
            borderRadius: const BorderRadius.all(Radius.circular(8)),
            selectedBorderColor: Colors.blue[700],
            selectedColor: Colors.white,
            fillColor: Colors.blue[200],
            color: Colors.blue[400],
            isSelected: _selectedWeather,
            children: icons,
          ),
        ],
      ),
    ),
  ),
  floatingActionButton: FloatingActionButton.extended(
    onPressed: () {
      setState(() {
        // When the button is pressed, ToggleButtons direction is changed.
        vertical = !vertical;
      });
    },
    icon: const Icon(Icons.screen_rotation_outlined),
    label: Text(vertical ? 'Horizontal' : 'Vertical'),
  ),
 );
}
```

```
}
```

Code 7 marks

Q.2. Attempt any Three **(21)**

a. Write a code and process to add image in a page (7)

```
import 'package:flutter/material.dart';

// function to start app building
void main() => runApp(const MyApp());

class MyApp extends StatelessWidget {
 const MyApp({Key? key}) : super(key: key);

// This widget is the root
// of your application

 @override
 Widget build(BuildContext context) {
  return MaterialApp(
   home: Scaffold(
    appBar: AppBar(
     title: const Text(
      'Insert Image Demo',
     ),
    ),
    body: Center(
     child: Column(
      children: <Widget>[
       Image.asset('assets/images/output.gif',
          height: 200,
          scale: 2.5,
          // color: Color.fromARGB(255, 15, 147, 59),
          opacity:
             const AlwaysStoppedAnimation<double>(0.5)), //Image.asset
       Image.asset(
        'assets/images/geeksforgeeks.jpg',
        height: 400,
        width: 400,
       ), // Image.asset
      ], //<Widget>[]
     ), //Column
    ), //Center
   ),
  );
 }
}
```

7 marks

b. Write a code to increase counter on tap of button

```
import 'package:flutter/material.dart';

void main() {
 runApp(const MyApp());
}

class MyApp extends StatelessWidget {
 const MyApp({Key? key}) : super(key: key);

 // This widget is the root of your application.
 @override
 Widget build(BuildContext context) {
  return MaterialApp(
   title: 'Flutter Demo',
   theme: ThemeData(

    primarySwatch: Colors.blue,
   ),
   home: const MyHomePage(title: 'Flutter Demo Home Page'),
  );
 }
}

class MyHomePage extends StatefulWidget {
 const MyHomePage({Key? key, required this.title}) : super(key: key);



 final String title;

 @override
 State<MyHomePage> createState() => _MyHomePageState();
}

class _MyHomePageState extends State<MyHomePage> {
 int _counter = 0;

 void _incrementCounter() {
  setState(() {

   _counter++;
  });
 }

 @override
 Widget build(BuildContext context) {
  return Scaffold(
   appBar: AppBar(
```

```
        title: Text(widget.title),
      ),
      body: Center(

        child: Column(

          mainAxisAlignment: MainAxisAlignment.center,
          children: <Widget>[
            const Text(
              'You have pushed the button this many times:',
            ),
            Text(
              '$_counter',
              style: Theme.of(context).textTheme.headline4,
            ),
          ],
        ),
      ),
      floatingActionButton: FloatingActionButton(
        onPressed: _incrementCounter,
        tooltip: 'Increment',
        child: const Icon(Icons.add),
      ), // This trailing comma makes auto-formatting nicer for build methods.
    );
  }
}
```

7 Marks

c. Write UI code to add image in background and to add two text feild and one toggle button and at the end one button (7)

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/background_image">

    <EditText
        android:id="@+id/text_field_1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Enter text 1 here"
        android:padding="16dp"
        android:layout_marginTop="100dp"/>

    <EditText
        android:id="@+id/text_field_2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Enter text 2 here"
        android:padding="16dp"
```

```
    android:layout_below="@id/text_field_1"/>

<ToggleButton
    android:id="@+id/toggle_button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Toggle button"
    android:layout_below="@id/text_field_2"
    android:layout_marginTop="16dp"
    android:layout_marginStart="16dp"/>

<Button
    android:id="@+id/submit_button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Submit"
    android:layout_alignParentBottom="true"
    android:layout_centerHorizontal="true"
    android:layout_marginBottom="32dp"/>

</RelativeLayout>
```

Code 7 Marks

d. Elaborate the light mode and Dark mode with the example and write down the precautions measures. (7)

Readability and Legibility – Let me explain the point scientifically. I am sure you will agree that the sun, planets, and other bodies emit electromagnetic energy of different wavelengths? A portion of that spectrum is visible to the human eye; of course, depending on the wavelength or the color of that light. For example, a clear sky is blue because clean air scatters blue light wavelengths more than red ones. In addition to this, wavelengths can boost our levels of attention, mood, and reaction time during the day. It's a different story at night, though. And that's the reason why e-book services, such as Google Play Books, introduce features such as "Night Light" for reading.

Accessibility – While creating a theme, keeping the contrast between the background, text colors, font size, and other accessibility factors is extremely important. After all, you are creating a project for people who aren't YOU, but for a broad range of users. To be precise, something that might have worked for one user might not work for another; a younger audience has different needs and preferences than an older audience. Apart from age, special needs, or disabilities, all of these can also determine the choice of color for the background and layout elements. Here the keyword is to research well to get closer to the target audience.

Clarity – in a general sense it is defined as the ability to see and distinguish all the core details on the screen or page. Let's take simplicity and intuitiveness of navigation for starters; if your end-user can easily find zones of information and elements of interaction just by scanning the layout, he or she doesn't need to put much effort in getting what they are looking for. In case the aspect is not tested properly; it might lead to weak visual hierarchy ending up in a complete mess.

Responsiveness – In layman's word, the responsiveness of the interface means that what users get is usable and functional irrespective of the device they prefer using.

For example, if something which looked stylish and appealing in Sketch on a high-res professional monitor may turn into a dirty stain on the small resolution screen. I am sure you must have encountered a situation where some color schemes looking nice at the design stage may lose their beauty in a variety of everyday conditions they are applied in. Provided the fact that a color scheme can have a significant impact on color, shape, and copy perceptions, one should test project on diverse devices and screens before making the final decision.
Environment – Last but certainly not least, is a question in which environments these web and mobile interfaces will be used. For example, for constant use under natural light, a dark background can literally create the effect of reflection, especially on glossy screens typical for tablets and smartphones. In a bad environment, a dark background can take the light away from the screen leading to bad influence on navigation and readability. In a nutshell, color combinations, contrast, and shades draw big attention here.
When to use dark UI?
When there is little text to read and more images/videos to watch
When there are very few elements in the design and are spaced well
When you want to give your users the feel of a dark environment like in a movie
When you want the core element of the page to stand out and gain prominence
When you want to reduce eye strain and make it comfortable for users to stay long in your website
When to use light UI?
When your website or app will be used mostly in the daytime
When your readers will be reading a lot of text
When there are a lot of elements on the screens
When you are using different color elements

Explanation 7 marks

Q.3. Attempt any Three **(21)**

a. Write a code to accept Name age address from the user, store the details in one JSON structure and save the data in JSON. (7)

```
class Person {
 String name;
 int age;
 String address;

 Person({required this.name, required this.age, required this.address});

 Map<String, dynamic> toJson() =>
    {'name': name, 'age': age, 'address': address};
}
class PersonForm extends StatefulWidget {
 @override
 _PersonFormState createState() => _PersonFormState();
}

class _PersonFormState extends State<PersonForm> {
```

```
final _formKey = GlobalKey<FormState>();
final _nameController = TextEditingController();
final _ageController = TextEditingController();
final _addressController = TextEditingController();

@override
void dispose() {
 _nameController.dispose();
 _ageController.dispose();
 _addressController.dispose();
 super.dispose();
}

@override
Widget build(BuildContext context) {
 return Form(
  key: _formKey,
  child: Column(
   crossAxisAlignment: CrossAxisAlignment.start,
   children: <Widget>[
    TextFormField(
     controller: _nameController,
     decoration: InputDecoration(
      labelText: 'Name',
     ),
     validator: (value) {
      if (value == null || value.isEmpty) {
       return 'Please enter your name';
      }
      return null;
     },
    ),
    TextFormField(
     controller: _ageController,
     decoration: InputDecoration(
      labelText: 'Age',
     ),
     keyboardType: TextInputType.number,
     validator: (value) {
      if (value == null || value.isEmpty) {
       return 'Please enter your age';
      }
      if (int.tryParse(value) == null) {
       return 'Please enter a valid age';
      }
```

```
            return null;
          },
        ),
        TextFormField(
          controller: _addressController,
          decoration: InputDecoration(
            labelText: 'Address',
          ),
          validator: (value) {
            if (value == null || value.isEmpty) {
              return 'Please enter your address';
            }
            return null;
          },
        ),
        Padding(
          padding: const EdgeInsets.symmetric(vertical: 16.0),
          child: ElevatedButton(
            onPressed: () {
              if (_formKey.currentState!.validate()) {
                final person = Person(
                  name: _nameController.text,
                  age: int.parse(_ageController.text),
                  address: _addressController.text,
                );
                saveData(person);
                ScaffoldMessenger.of(context).showSnackBar(
                  SnackBar(content: Text('Data saved')),
                );
              }
            },
            child: Text('Save'),
          ),
        ),
      ],
    ),
  );
}

void saveData(Person person) async {
  final prefs = await SharedPreferences.getInstance();
  final personJson = jsonEncode(person.toJson());
  await prefs.setString('person', personJson);
}
}
```

Code 7 marks

b. Elaborate following firebase authentication method (7)

Firebase Authentication is a service offered by Google's Firebase platform that provides developers with the ability to easily add user authentication and authorization to their mobile and web applications. Firebase Authentication supports several different authentication methods, including mobile, email, and custom authentication. Let's take a closer look at each of these methods:

Mobile Authentication: Mobile authentication is a method of authentication that allows users to sign in to an application using their phone number. This method is convenient for users who do not want to create and remember a separate username and password. Firebase Authentication uses SMS messages to verify the user's phone number, and once verified, the user can access the app. Mobile authentication is supported on both Android and iOS platforms.

Email Authentication: Email authentication is a widely used method that allows users to sign in to an application using their email address and a password. This method is secure and easy to use, as users can use their existing email accounts to authenticate themselves. Firebase Authentication provides email authentication out-of-the-box and allows developers to customize the sign-in flow. The service also supports passwordless authentication, which allows users to sign in to an app without a password by sending them a sign-in link via email.

Custom Authentication: Custom authentication is a method that enables developers to implement their own authentication system, which can be useful if the app requires specific user data or if the app has an existing user database. With custom authentication, developers can authenticate users using their own backend servers or other identity providers such as Facebook or Google. Firebase Authentication provides a way to authenticate users with custom tokens, which developers can generate and verify using Firebase Admin SDK.

In summary, Firebase Authentication provides several authentication methods, including mobile, email, and custom authentication, making it easy for developers to add user authentication and authorization to their apps. The service also provides a secure and scalable authentication infrastructure, allowing developers to focus on developing the core features of their apps.

c. Explain loader write code for loader. (7)

a loader is a widget that is commonly used to indicate to the user that a process or action is ongoing or in progress. The loader is often displayed as a spinning circular icon or progress bar, and it can be implemented using the CircularProgressIndicator widget

```
import 'package:flutter/material.dart';
Center(
  child: CircularProgressIndicator(),
),
Center(
  child: CircularProgressIndicator(
    valueColor: AlwaysStoppedAnimation<Color>(Colors.red),
  ),
),
```

```
Center(
 child: CircularProgressIndicator(
  valueColor: AlwaysStoppedAnimation<Color>(Colors.red),
  semanticsLabel: 'Loading...',
 ),
),
```

Explanation 2 Makrs 5 marks code

d. Elaborate how mobile and server can interact with each other and list the required permission. (7)

Mobile and server interaction is a fundamental component of many modern applications. The mobile app acts as the client, while the server provides the backend services and data storage required by the app. In order for the mobile app and server to communicate with each other, there are several mechanisms that can be used, such as APIs, HTTP requests, and socket connections.

Here's a high-level overview of how mobile and server interact with each other:

The mobile app sends a request to the server, such as a data query or a user action. This request is typically sent over a network connection, such as Wi-Fi or cellular data.

The server receives the request and processes it. This may involve retrieving data from a database, executing business logic, or performing other operations.

The server generates a response to the request, such as data to be displayed in the app or a status code indicating success or failure.

The server sends the response back to the mobile app over the network connection.

The mobile app receives the response and updates its UI or performs other actions based on the data or status code.

In order for the mobile app to communicate with the server, the app must have the necessary permissions and configurations. The required permissions depend on the type of communication protocol used and the security requirements of the application. Here are some examples of the required permissions:

Network access permission: The app must have permission to access the network, such as Wi-Fi or cellular data.

Internet permission: The app must have permission to use the internet to communicate with the server.

API key or authentication token: The app must have a valid API key or authentication token to access the server's services or data.

User permission: If the app requires access to the user's location, contacts, or other sensitive information, the user must grant permission to the app.

Firewall and security configurations: The server must have appropriate firewall and security configurations to prevent unauthorized access and protect the data and services provided by the server.

In summary, mobile and server interaction involves sending requests and responses over a network connection using APIs, HTTP requests, or socket connections. The mobile app must have the necessary permissions and configurations to communicate with the server, such as network access permission, API key or authentication token, and user permission. The server must have appropriate firewall and security configurations to protect its services and data.

Explanation 7 Marks

Q.4. Attempt any Three **(12)**

a. Explain pubspec.yaml file. (4)

In Flutter, the pubspec.yaml file is a configuration file that defines the dependencies, assets, and metadata for a Flutter app or package. This file is located in the root directory of the project and is used by the Flutter tool to manage dependencies, generate code, and build the app.

Here's a breakdown of the different sections and properties of the pubspec.yaml file:

name - This property specifies the name of the Flutter app or package. It should be a valid Dart package name and should not contain any spaces or special characters.

description - This property provides a brief description of the app or package.

version - This property specifies the version number of the app or package. It should follow the semantic versioning format (major.minor.patch) and should be updated whenever a new version is released.

environment - This section specifies the minimum version of the Flutter SDK required to run the app or package.

dependencies - This section specifies the external packages or libraries required by the app or package. Each dependency should be listed with its name and version number.

dev_dependencies - This section specifies the development dependencies required by the app or package, such as testing frameworks or code generators. These dependencies are not included in the final app or package.

flutter - This section specifies the Flutter-specific configurations for the app or package, such as the Flutter SDK version, assets, and plugins.

assets - This property specifies the list of asset files that should be included in the app or package. These can be images, fonts, or other files that are required by the app.

plugins - This property specifies the list of Flutter plugins that should be included in the app or package. These plugins provide additional functionality to the app, such as accessing device sensors or interacting with native APIs.

The pubspec.yaml file is an important configuration file in Flutter that defines the dependencies, assets, and metadata for a Flutter app or package. By managing the dependencies and configurations in this file, the Flutter tool can generate code, build the app, and package it for distribution.

Explanation 4 Marks

b. With the help of an example explain following (4)

1. Assest

In Flutter, assets are any files that are bundled and deployed with the app, such as images, fonts, audio or video files, and configuration files. These assets can be accessed by the app at runtime and used to provide content, style, or configuration to the app.

For example, let's say you have an image file named 'my_image.png' that you want to use in your Flutter app. To include this file as an asset, you would first add it to the 'assets' section of your pubspec.yaml file

2. Build.gradle Android, the build.gradle file is a configuration file that controls the build process for the app. It specifies the dependencies, build settings, and other configurations required to compile the app and package it for distribution.

For example, let's say you want to add a new library to your Android app. To do this, you would first add the library as a dependency in your app's build.gradle file:

Explnation 4 marks

c. Differentiate Firebase cloud firestore and Firebase RTDB (4)

Firebase Cloud Firestore and Firebase Realtime Database (RTDB) are both NoSQL databases offered by Firebase. While both databases are suitable for real-time data syncing and work well with Firebase's other services, there are some differences between them:

Data Structure:

Firebase RTDB stores data as a large JSON tree, where each key maps to a value. Firestore, on the other hand, stores data in a more structured format called collections and documents. In Firestore, each document contains fields, which are key-value pairs.

Querying:

Firestore supports more complex queries, such as range queries, sorting, and filtering by multiple fields. Firebase RTDB, in contrast, only supports simple queries, such as ordering and filtering by a single field.

Scalability:

Firestore is designed to handle larger and more complex datasets, while Firebase RTDB is better suited for smaller, simpler data models. Firestore also has better scalability features, such as sharding, which allows the database to be split across multiple servers.

Cost:

The cost of using Firestore is based on the number of reads, writes, and storage used, while Firebase RTDB is charged based on the amount of data transferred over the network.

Real-time updates:

Both Firestore and RTDB support real-time updates, but the way they handle these updates is different. Firebase RTDB pushes updates to clients as soon as they occur, while Firestore uses a more efficient mechanism called real-time listeners that only updates clients when the relevant data changes

Explanation 4 Marks

d. Differentiate Hybrid application development and native application development (4)

Hybrid application development and native application development are two approaches to developing mobile apps. Here are some of the key differences between the two:

Technology:

Native apps are developed using platform-specific technologies and languages such as Swift or Objective-C for iOS and Java or Kotlin for Android. Hybrid apps, on the other hand, are built using web technologies such as HTML, CSS, and JavaScript and are wrapped in a native container using frameworks such as React Native or Ionic.

Performance:

Native apps generally offer better performance and faster response times compared to hybrid apps. This is because native apps are optimized for the specific platform and can take advantage of the platform's hardware and software features, while hybrid apps are essentially web apps running in a container and can be slower and less responsive.

User experience:

Native apps can provide a better user experience by taking advantage of the platform's native features and user interface guidelines. Hybrid apps may have some limitations in this regard, as they may not be able to fully utilize all the native features of the platform.

Development speed:

Hybrid apps can be developed faster compared to native apps, as they can be built using web technologies and the same codebase can be used across multiple platforms. Native app development, on the other hand, requires separate development for each platform.

Cost:

Hybrid app development can be less expensive compared to native app development, as the same codebase can be used across multiple platforms, reducing the amount of development effort required. Native app development can be more expensive as it requires separate development for each platform.

Explanation 4 Marks

****