MAD CAT2 ANSWERS

a) Explain Layout & its types.

Q.1 a) Explain layout and its types

→ (i) Layout basically refers to the arrangement of elements on a page. These elements are likely to be buttons, texts or styles.

(ii) It is used to define the UI that holds UI controls or widgets that will appear on the screen of an android app.

(3) There are no. of layouts provided by android which you will use in almost all the android app. to provide different view, look and feel.

* Types: →

(1) Linear layout: It is a view group that align all children in a single direction, vertically or horizontally.

(2) Relative layout: It is a view group that display child view in 'relative positions.

(3) Table layout: It is a view that groups views into rows and columns.

(4) Absolute layout: It enables you to specify the exact location of its children.

(5) Frame layout: It is a placeholder on screen that you can use to display a single view.

(6) List view: It is a view group/that display a list of scrollable items.

(7) Grid view: It is a view group that display items in two-dimensional, scrollable grid.

b) Explain the procedure steps of Publishing Android App

→ Step 1: Sign up:

Sign up for an a/c on the Android Developer console, create an account cost $25.

Step 2: Create new application:

Select the publish an android Application option and fill out the details

Step 3: prepare multimedia:

Screenshots of app, Hi-res icon and feature graphic.

Step 4: prepare code for release:

- Remove log statements.
- Remove the android: debuggable attribute from your manifest file.
- Set the android: versionCode attribute in manifest-xml
- Set the android: versionName attribute in the manifest tag in manifest.xml

Step 5: Build a release-ready APK

The release-ready APK is different from the debug APK in that it is signed with certificate that is owned by the developer. Android Studio → Build → Generate Signed APK

A Java keystore (JKS) is a repository of public-private key pairs. You must sign all APKs with the same key pair.

Step 6: Upload APK:

Go back to the developer console and click on manage Releases. Then create a production Release and upload your signed APK.

Google will perform a check on the APK. If app was using an SHA for the launcher icon, which is no-bueno, then change it to PNG and recreate the signed APK.

Step 7:

Complete the checklist on the left until all the items have a green checkmark. The console re-evaluates the checklist every time you click Save Draft in the top right.

a) Describe Animation with their types

→ (1) Animations can add visual cues that notify users about what's going on in your app.

(2) They are especially useful when the UI changes state, such as when new content loads or new actions become available.

(3) Animations also add a polished look to your app, which gives it a higher quality look and feel.

(4) The animations are basically of three types as follows:

a) property Animation: →

It is one of the robust framework which allows animating almost everything. It can be used to add animation in checkbox, Radio buttons, and widgets other than any view.

b) View animation: →

It can be used to add animation to a specific view to perform tweened animation on views. Example can be seen in expandable Recycler View.

c) Drawable Animation: →

It is used if you want to animate one image over other.

(5) Important methods of Animations are:

a) startAnimation(): This method will start the animation.

b) clearAnimation(): This method will clear the animation running on a specific view.

b) What are the attributes for Layout

Layout Attributes

Each layout has a set of attributes which define the visual properties of that layout. There are few common attributes among all the layouts and their are other attributes which are specific to that layout. Following are common attributes and will be applied to all the layouts:

Sr.No	Attribute & Description
1	**android:id** This is the ID which uniquely identifies the view.
2	**android:layout_width** This is the width of the layout.
3	**android:layout_height** This is the height of the layout
4	**android:layout_marginTop** This is the extra space on the top side of the layout.
5	**android:layout_marginBottom** This is the extra space on the bottom side of the layout.
6	**android:layout_marginLeft** This is the extra space on the left side of the layout.
7	**android:layout_marginRight** This is the extra space on the right side of the layout.
8	**android:layout_gravity** This specifies how child Views are positioned.
9	**android:layout_weight** This specifies how much of the extra space in the layout should be allocated to the View.
10	**android:layout_x** This specifies the x-coordinate of the layout.
11	**android:layout_y** This specifies the y-coordinate of the layout.
12	**android:layout_width** This is the width of the layout.

13	**android:paddingLeft** This is the left padding filled for the layout.
14	**android:paddingRight** This is the right padding filled for the layout.
15	**android:paddingTop** This is the top padding filled for the layout.
16	**android:paddingBottom** This is the bottom padding filled for the layout.

a) Define Explain User Interface screen Elements.

Q1 b)

→ i) A View is an object that draws something on the screen that the user can interact with and a ViewGroup is an object that holds other View objects in order to define layout of UI.

ii) There are no. of UI control provided by android that allow you to build the graphical UI :

a) TextView : This control is used to display text to the user.

b) Edit text : It is predefined subclass of TextView that includes rich editing capabilities.

c) Datepicker : The Datepicker view enables users to select a date of the day.

d) Timepicker : It view enables user's to select a time of day.

e) Spinner : A drop-down list that allow user to select one value from a set.

f) ProgressBar : It provides visual feedback about some ongoing task.

g) RadioButton : It has two states : either checked or unchecked.

---

h) RadioGroup : It is used to group together one or more RadioButton.

i) ToggleButton : An on/off button with light indicator

j) Button : A push-button that can be pressed, or clicked by the user to perform an action.

b) Write a Short Note On :

1. Sqlite with queries on CRUD

→ (1) Sqlite is a open source SQL database that stores data in the form of file on a device.

(2) Sqlite supports all the relational db features to store data permanently.

(3) CRUD is nothing but an abbreviation for the basic operation that perform in any database. And the operations are create, Read, update, delete.

(4) Query for creating a table :→

```
CREATE TABLE employees (
    id INTEGER NOT NULL CONSTRAINT employees_pk
    PRIMAY KEY AUTOINCREMENT,
    name Varchar (200) NOT NULL,
    department Varchar (200) NOT NULL,
    'joiningdate' datetime NOT NULL,
    Salary double NOT NULL
);
```

(5) Creating a new Record :→

```
INSERT INTO employees
(name, department, 'joiningdate', Salary)
VALUES
('Belal', 'Technical', '2017-09-3' 10:00:00', '40000');
```

(6) Reading All existing Records :→

```
SELECT * FROM employees;
```

⑦ Reading specific Record :→

```
SELECT * FROM employees WHERE id=1 ;
```

⑧ Updating a Record :→

```
UPDATE employees
SET name = 'Belal Haque',
    department = 'Research and Development',
    salary = '100000'
WHERE id=1 ;
```

a) Discuss Android APIs, explain their types

Q.3 a)

→ * APIs: →

① Application program interface (API) is a code for a programmer that they use in their applications.

② This code or (API) allows you to add specific functionalities to your application.

③ In otherword, we can say that APIs are the set of protocols and tools used for building s/w application.

* Type: →

There are four main types of APIs:

① Open APIs: Also known as public API, there are no restrictions to access these types of APIs because they are publicly available.

② Partner APIs: A developer needs specific rights or licenses in order to access this type of API because they are not available to the public.

③ Internal APIs: Also known as private APIs, only internal systems expose this types of API. These are usually designed for internal use within a company.

④ Composite APIs: This type of API combines different data and service APIs. Its main use is to speed up the process of execution and improve the performance.

b) Explain basic type of Android Testing

There are several types of testing that can be performed on Android applications. Here are some of the basic types of Android testing:

Unit testing: This involves testing individual units of code in isolation to ensure that they function as expected. Unit tests can be automated, and are typically run frequently throughout the development process.

Integration testing: This involves testing how different components of an application work together. Integration testing is used to verify that different parts of an application can communicate with each other and work correctly when combined.

Functional testing: This involves testing the functionality of the application as a whole. Functional tests are typically manual tests that are run by human testers to ensure that the application behaves as expected and meets the requirements.

Performance testing: This involves testing the performance of the application under different conditions, such as with different numbers of users or on different devices. Performance testing is used to identify bottlenecks and other issues that can affect the performance of the application.

Acceptance testing: This involves testing the application to ensure that it meets the requirements and specifications provided by the client or stakeholders. Acceptance testing is typically done near the end of the development process, and is used to ensure that the application is ready for release.

Q6) Explain the following w.r.t android testing

① manual testing :→

- Manual testing testing involves human interaction with app under different conditions or situations to analyze how it responds.
- A human performs the tests step by step without test scripts.
- Manual testing is the process in which QA analysts execute test one-by-one in an individual manner.
- The purpose of manual testing is to catch bugs and features issues before a site application goes live.
- manual testing should be used to perform

Page

exploratory testing, usability testing and Ad-hoc testing to exhibits the best results.

② Automated Testing :→

- Android automated testing requires no human interaction as code is automatically executed and reported.
- Tests are executed automatically via test automation frameworks, along with other tools and site.
- Automation testing should be used to perform Regression testing, load testing, performance testing and Repeated execution for best results.
- Automation testing is used to increase the efficiency, effectiveness and coverage.

a) Explain the tools of Android Testing

APPIUM

Being an open-source test automation framework, Appium enables testing teams to create tests for mobile web, hybrid and native applications without the need to recompile or alter your code. Working on the client-server architecture. Its server is written in Node.js, and it is compatible with popular client libraries like Ruby, Java, Python, PHP, and a few others.

KEY FEATURES:

It supports multiple languages that simplifies the process of creating test cases – your team can use their preferred programming language for writing test scripts.
It allows testing teams to execute tests against multiple mobile platforms.

SELENDROID

If you have a focus on developing Android apps, then Selendroid is for you. Utilized for both native and hybrid applications, it opens up ample opportunities for simultaneous automated testing of one app on several devices. Additionally, it can be integrated as a node into the Selenium Grid for scaling and parallel testing.

KEY FEATURES:

It offer UI testing – it inspects UI elements of the app being tested, even the oldest versions.
It provides the "hot plugging" feature that enables QA engineers to connect or disconnect different Android devices without interrupting the test being executed.
It doesn't need any modification for test automation

All-In-One Testing Solutions

KATALON

Powered with a Selenium engine, Katalon focuses on building and reusing automated UI test scripts without the need for coding. With a powerful recording utility, you can just reuse UI elements to speed up the testing process. Additionally, charts, graphs, and reports allow your Android testing team to visualize test data and test execution results to make any changes or further improvements.

KEY FEATURES:

It offers a recording and playback feature.
It provides a manual mode so that non-programmers are able to create automation test cases successfully as well as a script option for programmers.
It operates on both cloud and on-premise infrastructures.

It offers built-in integration to be connected with other tools like JIRA.

RANOREX

With a mix of recorded actions and drag-and-drop actions, testers can create test cases from a preset list or repository. Additionally, they are able to take screenshots at any point of the testing process which helps verify elements of the web applications.

KEY FEATURES:

It allows teams to create automated UI tests with a simple drag-and-drop feature thereby providing UI testing.

It provides teams with the ability to execute tests on multiple devices simultaneously.

It offers seamless integration with other tools and includes an object recognition feature to automatically identify any change in the UI.

Open-Source Mobile Testing Tool

MONKEY TALK

Available for free, this tool provides automation testing of native and hybrid software products. If you want to use it professionally, you should purchase this tool. However, there is no need to have solid programming or scripting knowledge to perform test automation.

KEY FEATURES:

It automates functional interactive tests for Android-based applications.

It includes record and playback functional test suites.

It easily captures the touch and gesture-based operations.

ROBOT FRAMEWORK

With an easy-to-use syntax, even non-technicians can automate tests using keywords. Thanks to this keyword-driven approach, testing teams can generate more test cases and increase productivity. All you need – just load the necessary libraries, such as Selenium Library, and create a test in this testing framework.

KEY FEATURES:

It is highly extensible with Python or Java libraries.

It generates detailed reports.

It is compatible with external libraries.

Android Studio’s Testing Tools

UI Automator

Being a Google-based testing framework, it supports Java and Kotlin and allows teams to efficiently perform black-box testing. Powered with a UI Automator Viewer, it allows testers to interact with visible elements on a device.

Key Features:

It includes APIs that support cross-app UI testing and allow to retrieve the state information and perform operations on the target device.

When creating tests, there is no need for the test engineers to know about the implementation details of the app.

Espresso

Built by Google, Espresso is utilized as a native testing framework to perform automated testing for Android-based products. Being a part of the Android SDK, QA teams apply it to test native mobile apps. With its flexible API, testing engineers can not only create tests that are close to the Android app's logic but also write black box tests.

Key features:

It provides a highly stable test cycle.

It offers easy-to-learn and very simple API.

It enables automatic synchronization between your app and tests.

Testing Emulators

Android Studio

Designed to simplify the testing process, Android Studio is equipped with an embedded emulator that delivers a wide range of testing options. Whether you want to create, run, and analyze tests or a specific group of tests – it's a place to stay. What's more, teams can organize test execution on the local machine or run tests on a device.

Key features:

It allows you to test an app across multiple devices.

It provides the ability to simulate several different hardware features, including GPS, multiple touch inputs, etc.

It allows teams to visualize test results.

BlueStacks

Applied as an emulator for Android-based apps, BlueStacks comes with a ton of options to improve the app's experience. It offers superior performance and helps teams to run multiple Android instances smoothly.

KEY FEATURES:

It allows teams run multiple apps simultaneously.
It allows testing teams to record and replay any action in real time.
It enables testers to do video recordings and screen recordings.

ARCHON

Integrated as a Chrome extension, ARChon offers easy installation and enables to run the Android-based apps and games. It allows QA teams to simulate different device configurations, features, and even the ARCore platform. Additionally, external Android devices such as tablets, cameras, and other mobile gadgets can be easily added to meet testing needs.

KEY FEATURES:

It provides great customization and support for the latest Android system.
It supports Google Chrome OS.

MEMU PLAY

This emulator is considered to be an ideal tool for testing high-end games. It is lightweight and doesn't need too much space to be installed.

KEY FEATURES:

It is very convenient for usage.
It provides great performance for integrated graphics of the app.
It offers virtualization and a custom key mapping option.

AI-Powered Software

EGGPLANT

With a focus on app testing and GUI testing, this tool helps testers thoroughly test mobile apps by creating multiple device scenarios. Thanks to image analysis technology, it extracts text from images and validates complex UX elements. What's more, this AI-driven tool allows you to predict how an app will behave under various conditions.

KEY FEATURES:

It supports the entire Android testing process, from test case generation to results analytics.
It provides integration with application lifecycle management software.

It adapts test models automatically to the UI testing needs.

TestComplete

TestComplete allows QA engineers to create automated tests in various scripting languages for a great number of apps. With its flexible UI, they can run functional tests quickly. However, the most important thing is executing parallel regression tests with automation builds and creating stable regression tests.

Key features:

It is easy to use for technical and non-technical team members.

It scales with your Android project.

some of the commonly used tools for Android testing:

Android Studio: Android Studio is the official Integrated Development Environment (IDE) for Android development, and it comes with a built-in testing framework called Android Test. Android Studio supports different types of testing, including unit testing, integration testing, and UI testing, and allows developers to run tests on different devices and emulators.

Espresso: Espresso is a testing framework that allows developers to write UI tests for Android applications. Espresso provides a set of APIs that enable developers to interact with UI components, such as buttons and text fields, and verify the app's behavior. Espresso tests can be run on emulators or real devices, and they can be integrated with CI/CD tools such as Jenkins or Travis CI.

UI Automator: UI Automator is another testing framework that allows developers to write UI tests for Android applications. It provides a set of APIs that enable developers to interact with UI components across different applications, such as pressing buttons, entering text, or swiping screens. UI Automator tests can also be run on emulators or real devices.

Robolectric: Robolectric is a testing framework that allows developers to run unit tests for Android applications without the need for an emulator or device. Robolectric provides a simulated Android environment, allowing developers to test their code's behavior without the overhead of running on a physical device or emulator.

Mockito: Mockito is a mocking framework that allows developers to create mock objects for testing purposes. Mock objects can simulate the behavior of real objects and allow developers to test their code's behavior in isolation. Mockito is commonly used in combination with other testing frameworks such as JUnit or Espresso.

Firebase Test Lab: Firebase Test Lab is a cloud-based testing service provided by Google. It allows developers to run tests on different configurations of Android devices, such as different versions of Android or different screen resolutions. Firebase Test Lab provides a web interface for creating test configurations and viewing test results.

These are just a few examples of the tools available for Android testing, and there are many other options available depending on your testing needs. By using a combination of these tools, developers can ensure that their Android applications are high-quality, perform well, and meet the requirements of their users.

b) Explain View & ViewGroup

In Android, both View and ViewGroup are classes used to create the user interface (UI) components of an application.

A View represents a single UI component, such as a TextView, EditText, Button, or ImageView. It is the basic building block of any Android UI and is responsible for drawing and handling user interaction for a specific area on the screen. Each View has a unique identifier, which can be used to access and manipulate it programmatically.

A ViewGroup, on the other hand, is a special type of View that contains other Views. It can be thought of as a container for other Views and is used to group related UI elements together. ViewGroup is an abstract class, and there are several concrete classes that extend it, such as LinearLayout, RelativeLayout, FrameLayout, and ConstraintLayout.

When a ViewGroup contains other Views, it is responsible for laying out those Views on the screen. The ViewGroup uses a layout manager to arrange the child Views according to specific rules or constraints. The layout manager can be set to position the child Views in a horizontal or vertical direction, relative to each other, or to fill the available space.

In summary, a View represents a single UI component, and a ViewGroup is a container for other Views. Together, they form the foundation of Android UI development, allowing developers to create rich, interactive, and dynamic user interfaces.

a) Write a short note on :APIs

Q.3 a)

→ * APIs : →

① Application program interface (API) is a code for a programmer that they use in their applications.

② This code or (API) allows you to add specific functionalities to your application.

③ In otherword, we can say that APIs are the set of protocols and tools used for building s/w application.

* Type : →

There are four main types of APIs :

① Open APIs : Also known as public API, there are no restrictions to access these types of APIs because they are publicly available.

② Partner APIs : A developer needs specific rights or licenses in order to access this type of API because they are not available to the public.

③ Internal APIs : Also known as private APIs, only internal systems expose this types of API. They are usually designed for internal use within a company.

④ Composite APIs : This type of API combines different data and service APIs. Its main uses are to speed up the process of execution and improve the performance.

API stands for Application Programming Interface. It is a set of protocols, tools, and routines that software developers use to build and integrate different software applications. APIs allow applications to communicate with each other and exchange data in a standardized and secure way.

In the context of the internet, APIs are commonly used to integrate web-based applications with other applications or services. For example, social media platforms use APIs to allow third-party developers to integrate their applications with the platform's features, such as posting updates or retrieving user data.

APIs come in different types, such as web APIs, operating system APIs, or library APIs. Web APIs are designed to interact with web-based applications and are typically accessed through HTTP requests. Operating system APIs allow applications to interact with the underlying operating system, while library APIs provide a set of functions or classes that can be used by other applications.

APIs are essential in modern software development, as they enable developers to build more complex and integrated applications. APIs also provide a standardized way of communicating between different applications, reducing the complexity of software development and making it easier to integrate with other services.

a) Write a short note on
1) Toast Notification

Toast notification is a small message that appears on the screen of an Android device for a short period of time. It is a simple way to provide feedback to the user and to display information that doesn't require immediate attention.

Toast notifications are commonly used in Android applications to display brief messages, such as confirmation messages, error messages, or status updates. They are displayed in a small rectangular box at the bottom of the screen and disappear after a few seconds.

To create a Toast notification, developers can use the Toast class provided by the Android SDK. The class allows developers to specify the text to be displayed, the duration of the message, and the location on the screen where the message should appear.

Toast notifications are a convenient way to provide quick feedback to users without interrupting their workflow or requiring them to take any action. They are simple to implement and can be customized to match the look and feel of the application. However, developers should use Toast notifications sparingly and avoid displaying too many messages, as this can be distracting and annoying for users.

2) Toggle Button

A ToggleButton is a UI element in Android that represents a two-state button. It is a subclass of CompoundButton and inherits its properties and methods.

The two states of a ToggleButton are represented by the on/off states of the button. The button has a text label that can be set for each state, and the user can toggle between the two states by tapping on the button.

ToggleButton can be used in a variety of scenarios, such as enabling or disabling a feature, switching between two modes, or selecting an option from a set of choices.

To use a ToggleButton in an Android application, developers can add it to the layout XML file and set its properties, such as the text labels for the on/off states, the initial state, and any listeners to handle state changes.

When the user taps on the button, the state of the ToggleButton changes and an event is generated. Developers can handle this event to perform any necessary actions based on the new state of the button.

ToggleButton is a simple and easy-to-use UI element that can be used to provide an intuitive way for users to toggle between two states or options in an Android application.

a) Describe the process of mobile application for integration with GPS.

1. Determine the requirements: The first step is to determine the requirements for GPS integration in the application. This includes identifying the types of location data needed, such as real-time location updates, distance traveled, or geofencing.
2. Choose a GPS provider: There are several GPS providers available that offer APIs for developers to integrate into their mobile applications. The choice of provider will depend on the specific requirements of the application.
3. Integrate the GPS API: Once a GPS provider is selected, the next step is to integrate the GPS API into the mobile application. This typically involves registering the application with the GPS provider, obtaining an API key, and configuring the API to receive location data.
4. Request location updates: To receive location updates from the GPS, the application needs to request permission from the user to access their device's location services. Once permission is granted, the application can use the GPS API to request location updates at specified intervals.
5. Process location data: The application can process the location data received from the GPS to perform various functions, such as displaying the user's location on a map, calculating distance traveled, or triggering actions based on geofencing.
6. Optimize battery usage: To optimize battery usage, the application can use various techniques, such as reducing the frequency of location updates, using low-power modes, or detecting when the device is idle.
7. Test and refine: Finally, the application should be thoroughly tested to ensure that it is working as expected. Any issues or bugs should be identified and addressed before the application is released to users.

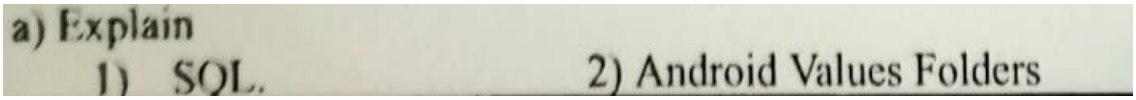
b) What are some Do's and Don'ts in the android market ?

- Do's:

1. Do research on your target audience before launching your app in the Android market. This will help you understand their needs and preferences, and tailor your app accordingly.
2. Do make sure that your app is well-designed and user-friendly. A good user experience is crucial for the success of your app.
3. Do provide regular updates to your app to fix bugs and add new features. This will keep your users engaged and improve your app's ratings and reviews.
4. Do take advantage of the various promotional tools available in the Android market, such as app store optimization, paid advertising, and social media marketing.
5. Do respond to user feedback and address any issues or concerns promptly. This will help you build a loyal user base and improve your app's reputation.

- Don'ts:

1. Don't launch your app without proper testing. Ensure that your app is stable and bug-free before releasing it to the public.
2. Don't violate any of the Google Play Store policies or guidelines. This can result in your app being removed from the market, and damage your reputation as a developer.
3. Don't overload your app with unnecessary features or functionalities. This can make your app complex and confusing for users.
4. Don't neglect your app's marketing and promotion. A well-marketed app is more likely to be discovered by users and achieve higher downloads and ratings.
5. Don't ignore user feedback or criticism. Listen to your users and make necessary improvements to your app to ensure its success in the Android market.



SQL is a language to operate databases; it includes Database Creation, Database Deletion, Fetching Data Rows, Modifying & Deleting Data rows, etc.

SQL stands for **Structured Query Language** which is a computer language for storing, manipulating and retrieving data stored in a relational database.

SQL is widely popular because it offers the following advantages –

- Allows users to access data in the relational database management systems.
- Allows users to describe the data.
- Allows users to define the data in a database and manipulate that data.
- Allows to embed within other languages using SQL modules, libraries & pre-compilers.
- Allows users to create and drop databases and tables.
- Allows users to create view, stored procedure, functions in a database.
- Allows users to set permissions on tables, procedures and views.

There are five types of SQL commands: DDL, DML, DCL, TCL, and DQL :

1. Data Definition Language (DDL)

- DDL changes the structure of the table like creating a table, deleting a table, altering a table, etc.
- All the command of DDL are auto-committed that means it permanently save all the changes in the database.

Here are some commands that come under DDL:

- CREATE : It is used to create a new table in the database.
- ALTER : It is used to alter the structure of the database. This change could be either to modify the characteristics of an existing attribute or probably to add a new attribute.
- DROP : It is used to delete both the structure and record stored in the table.
- TRUNCATE : It is used to delete all the rows from the table and free the space containing the table.

2. Data Manipulation Language

- DML commands are used to modify the database. It is responsible for all form of changes in the database.
- The command of DML is not auto-committed that means it can't permanently save all the changes in the database. They can be rollback.

Here are some commands that come under DML:

- INSERT : The INSERT statement is a SQL query. It is used to insert data into the row of a table.
- UPDATE : This command is used to update or modify the value of a column in the table.
- DELETE : It is used to remove one or more row from a table.

3. Data Control Language

DCL commands are used to grant and take back authority from any database user.

Here are some commands that come under DCL:

- Grant : It is used to give user access privileges to a database.
- Revoke : It is used to take back permissions from the user.

4. Transaction Control Language

TCL commands can only use with DML commands like INSERT, DELETE and UPDATE only.

These operations are automatically committed in the database that's why they cannot be used while creating tables or dropping them.

Here are some commands that come under TCL:

- COMMIT : Commit command is used to save all the transactions to the database.
- ROLLBACK : Rollback command is used to undo transactions that have not already been saved to the database.
- SAVEPOINT : It is used to roll the transaction back to a certain point without rolling back the entire transaction.

5. Data Query Language

DQL is used to fetch the data from the database.

It uses only one command:

- SELECT : This is the same as the projection operation of relational algebra. It is used to select the attribute based on the condition described by WHERE clause.

b) Discuss the need for permissions in Android. Describe the permissions to set system functionalities like bluetooth,camera.

Permissions in Android refer to the security mechanism that controls access to specific resources, functions, or data on a user's device. Android applications require certain permissions to access device features, such as the camera, microphone, contacts, and location data.

The need for permissions in Android is essential for protecting user privacy and security. Without proper permissions, an application could potentially access sensitive data without the user's consent or knowledge. For example, an application could read and upload the user's contacts, location data, or personal information.

By requesting permissions, Android applications are required to inform the user of the specific resources or data that they intend to access. This allows users to make informed decisions about whether or not to grant access to the application. Permissions are granted on a per-application basis and can be managed by the user through the Android system settings.

In addition to protecting user privacy and security, permissions also provide developers with a standardized way of accessing device features and data. This makes it easier for developers to create applications that are consistent across different Android devices and versions.

Permissions play a critical role in ensuring the security and privacy of Android users. They provide a necessary layer of protection against malicious or unwanted access to sensitive data and resources on a user's device.

In Android, permissions are required for accessing system functionalities such as Bluetooth and camera. The specific permissions required for accessing these functionalities are as follows:

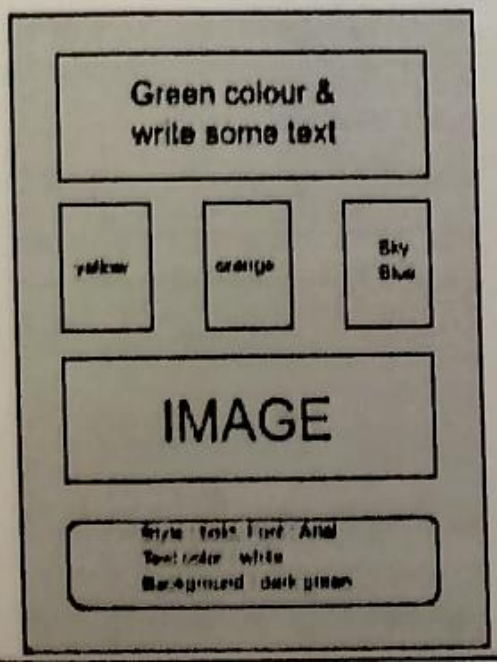
Bluetooth: The permission required for accessing Bluetooth functionality is "android.permission.BLUETOOTH". This permission is required for scanning nearby Bluetooth devices, connecting to paired Bluetooth devices, and communicating with Bluetooth devices.

Camera: The permission required for accessing the camera functionality is "android.permission.CAMERA". This permission is required for capturing photos or videos using the device's camera, and accessing the camera's preview frame data.

To request these permissions, an Android application must declare them in its AndroidManifest.xml file. When the application is installed, the user is prompted to grant or deny the requested permissions. If the user grants the permissions, the application is allowed to access the specified system functionality.

It is important to note that some system functionalities require additional permissions, depending on the specific use case. For example, accessing the microphone for recording audio requires the "android.permission.RECORD_AUDIO" permission.

a) Observe the following GUI and write an XML file using relative layout to create the same.



1. <? xml version = "1.0" encoding = "utf-8" ?>
< androidx.constraintlayout.widget.constraintlayout
2. xmlns: android = "http://schemas.android.com/apk/res/android"
3. xmlns: app = "https://schemas.android.com/apk/res-auto"
4. xmlns: tools = "http://schemas.android.com/tools"
5. android: layout_width = "match_parent"
6. android: layout_height = "match_parent"
7. android: background = "drawable/fourth"
8. android: visibility = "visible"
9. tools: context = ".mainActivity"
10. tools: visibility = "visible" >

< EditText
android: id = "@+id/myTextBox"
android: layout_width = "336 dp"
android: layout_height = "101 dp"
android: background = "#00FF00"
android: text = "Hello Everyone"
app: layout_constraintBottom_toBottomOf = "parent"
app: layout_constraintEnd_toEndOf = "parent"
app: layout_constraintHorizontal_bias = "0.115"
app: layout_constraintStart_toStartOf = "parent"
app: layout_constraintTop_toTopOf = "parent"
app: layout_constraintVertical_bias = "0.317" />

<TextView

android :id = "@+id /myBox1"

android : layout_width = "91 dp"

android : layout_height = "95 dp"

android : background = "@ color /white"

android : gravity = "center"

android : text = "@ string /yellow"

android : textcolor = "#827717"

android : visibility = "visible"

app : layout_constraintBottom_toBottomOf = "parent"

app : layout_constraintEnd_toEndOf = "parent"

app : layout_constraintHorizontal_bias = "0.115"

app : layout_constraintStart_toStartOf = "parent"

app : layout_constraintTop_toTopOf = "parent"

app : layout_constraintVertical_bias = "0.317"

tools : visibility = "visible" />

b) Write a short note on
1) SQLite

→ (1) Sqlite is a open source SQL database that stores data in the form of file on a device.
(2) Sqlite supports all the relational db features to store data permanently.
(3) CRUD is nothing but an abbreviation for the basic operation that perform in any database. And the operations are
create, Read, update, delete.
(4) Query for creating a table :→

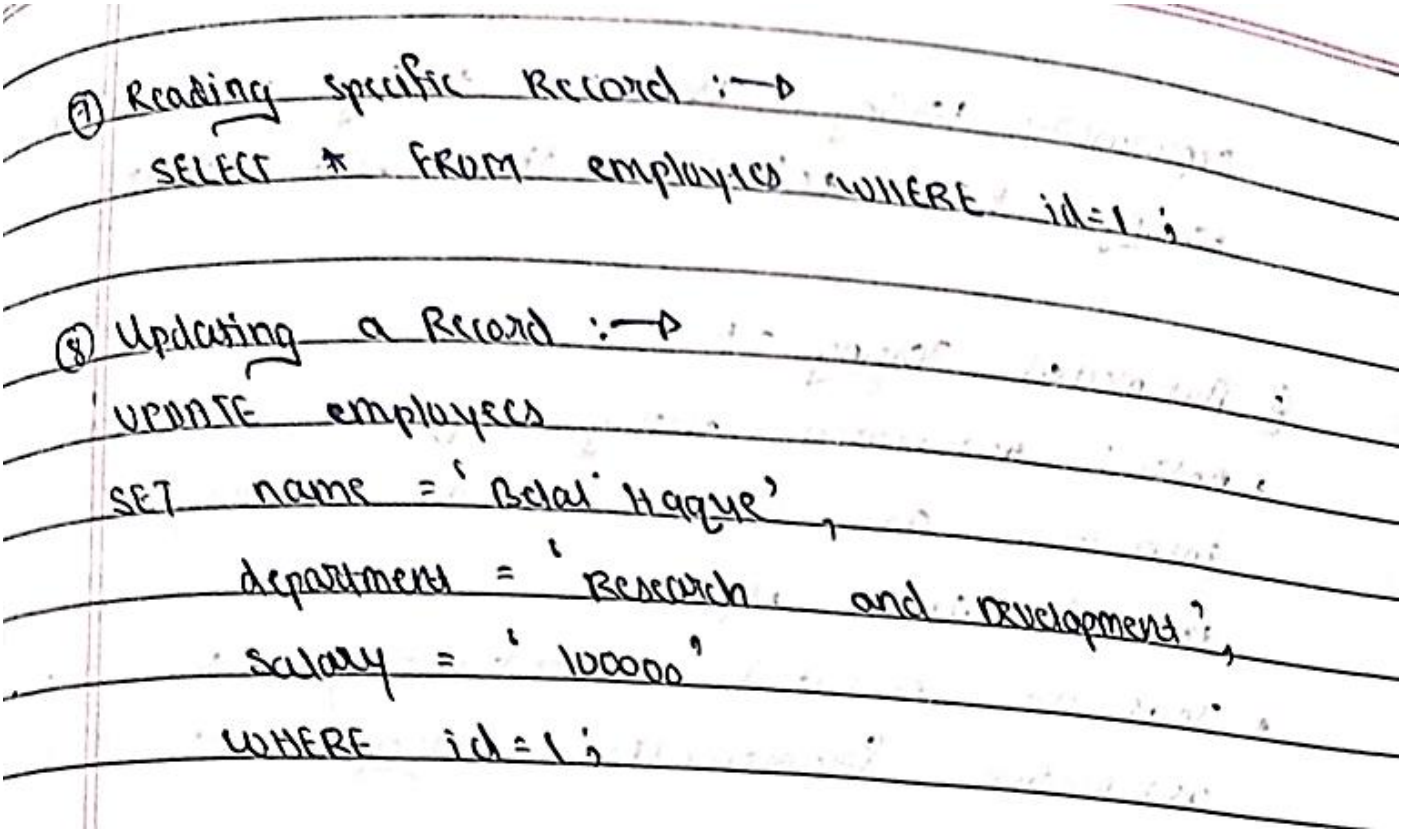
```
CREATE TABLE employees (
    id INTEGER NOT NULL CONSTRAINT employees_pk
    PRIMARY KEY AUTOINCREMENT,
    name varchar (200) NOT NULL,
    department varchar (200) NOT NULL,
    'joiningdate' datetime NOT NULL,
    salary double NOT NULL
);
```

(5) Creating a new Record :→

```
INSERT INTO employees
(name, department, 'joiningdate', salary)
VALUES
('Belal', 'Technical', '2017-09-3' 10:00:00', '40000');
```

(6) Reading All existing Records :→

```
SELECT * FROM employees;
```



SQLite is a popular open-source relational database management system (RDBMS) that is embedded in many mobile devices, including Android smartphones and tablets. It is a lightweight and self-contained RDBMS that requires minimal setup and administration.

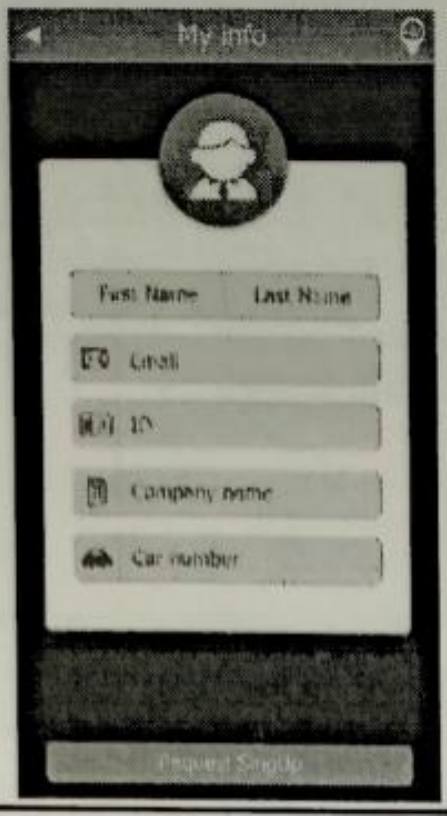
SQLite is the default database management system used in Android applications. It is a lightweight and efficient relational database that is designed to be embedded in mobile devices. SQLite databases in Android are used to store and manage structured data such as user profiles, settings, and application data.

SQLite supports standard SQL commands for creating, modifying, and querying databases. It is designed to be highly efficient and can handle large amounts of data with minimal impact on device performance. SQLite databases are stored as files on the device's file system, making them easy to manage and backup.

One of the key benefits of SQLite is its small footprint, which makes it ideal for use in mobile applications where storage and memory resources are limited. It is also highly reliable and offers transaction support, which helps to ensure data consistency and integrity.

Another benefit of SQLite is its portability, as databases can be easily transferred between devices and platforms. This makes it easy for developers to create applications that are consistent across different devices and operating systems.

a) Develop the registration form using the following GUI.



b) Develop a simple calculator using table layout.

The layout code snippet is shown below.

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.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="com.jackrutorial.calculatorexample.MainActivity">

   <TableLayout
      android:layout_width="match_parent"
      android:layout_height="wrap_content">

      <TableRow
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         >

         <TextView
            android:id="@+id/txtResult"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_span="4"
            android:layout_gravity="right"
            />

      </TableRow>

      <TableRow
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         >

         <EditText
            android:id="@+id/edtInput"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_span="4"
            android:layout_gravity="right"
            android:inputType="number"
            />

      </TableRow>
```

```
    <TableRow
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      >

      <Button
        android:id="@+id/btnCE"
        android:text="CE"
        />

      <Button
        android:id="@+id/btnC"
        android:text="C"
        />

      <Button
        android:id="@+id/btnDelete"
        android:text="Delete"
        android:layout_span="2"

        />

    </TableRow>

    <TableRow
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      >

      <Button
        android:id="@+id/btnNumber7"
        android:text="7"
        />

      <Button
        android:id="@+id/btnNumber8"
        android:text="8"
        />

      <Button
        android:id="@+id/btnNumber9"
        android:text="9"
        />

      <Button
        android:id="@+id/btnNumberAdd"
        android:text="+"
```

```
        />

    </TableRow>

    <TableRow
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      >

      <Button
        android:id="@+id/btnNumber4"
        android:text="4"
        />

      <Button
        android:id="@+id/btnNumber5"
        android:text="5"
        />

      <Button
        android:id="@+id/btnNumber6"
        android:text="6"
        />

      <Button
        android:id="@+id/btnNumberSub"
        android:text="-"
        />

    </TableRow>

    <TableRow
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      >

      <Button
        android:id="@+id/btnNumber1"
        android:text="1"
        />

      <Button
        android:id="@+id/btnNumber2"
        android:text="2"
        />

      <Button
        android:id="@+id/btnNumber3"
```

```
            android:text="3"
            />

        <Button
            android:id="@+id/btnNumberMul"
            android:text="*"
            />

    </TableRow>

    <TableRow
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        >

        <Button
            android:id="@+id/btnNumber0"
            android:text="0"
            />

        <Button
            android:id="@+id/btnDot"
            android:text="."
            />

        <Button
            android:id="@+id/btnResult"
            android:text="="
            />

        <Button
            android:id="@+id/btnNumberDiv"
            android:text="/"
            />

    </TableRow>

  </TableLayout>

</android.support.constraint.ConstraintLayout>
```

Main Activity

Change MainActivity activity code to the following.

```
package com.jackrutorial.calculatorexample;
```

```
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {

    Button btnNumber0;
    Button btnNumber1;
    Button btnNumber2;
    Button btnNumber3;
    Button btnNumber4;
    Button btnNumber5;
    Button btnNumber6;
    Button btnNumber7;
    Button btnNumber8;
    Button btnNumber9;

    TextView txtResult;

    EditText edtInput;

    Button btnCE;
    Button btnC;
    Button btnDelete;

    Button btnAdd;
    Button btnSub;
    Button btnMul;
    Button btnDiv;

    Button btnDot;
    Button btnResult;

    double val1=Double.NaN;
    double val2;
    String ACTION;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        btnNumber0 = (Button) findViewById(R.id.btnNumber0);
        btnNumber1 = (Button) findViewById(R.id.btnNumber1);
        btnNumber2 = (Button) findViewById(R.id.btnNumber2);
```

```
        btnNumber3 = (Button) findViewById(R.id.btnNumber3);
        btnNumber4 = (Button) findViewById(R.id.btnNumber4);
        btnNumber5 = (Button) findViewById(R.id.btnNumber5);
        btnNumber6 = (Button) findViewById(R.id.btnNumber6);
        btnNumber7 = (Button) findViewById(R.id.btnNumber7);
        btnNumber8 = (Button) findViewById(R.id.btnNumber8);
        btnNumber9 = (Button) findViewById(R.id.btnNumber9);

        txtResult = (TextView) findViewById(R.id.txtResult);

        edtInput = (EditText) findViewById(R.id.edtInput);

        btnCE = (Button) findViewById(R.id.btnCE);
        btnC = (Button) findViewById(R.id.btnC);
        btnDelete = (Button) findViewById(R.id.btnDelete);

        btnAdd = (Button) findViewById(R.id.btnNumberAdd);
        btnSub = (Button) findViewById(R.id.btnNumberSub);
        btnMul = (Button) findViewById(R.id.btnNumberMul);
        btnDiv = (Button) findViewById(R.id.btnNumberDiv);

        btnDot = (Button) findViewById(R.id.btnDot);
        btnResult = (Button) findViewById(R.id.btnResult);

        btnCE.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                edtInput.setText(null);
            }
        });

        btnC.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                val1 = Double.NaN;
                txtResult.setText(null);
                edtInput.setText(null);
            }
        });

        btnDelete.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String number = edtInput.getText().toString();
                if(number != null && number.length() > 0){
                    number = number.substring(0, number.length() - 1);
                }
                edtInput.setText(number);
```

```
            edtInput.setSelection(edtInput.getText().length());
        }
    });

    btnAdd.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            ACTION = "+";
            if(!Double.isNaN(val1)){
                val2 = Double.parseDouble(edtInput.getText().toString());
                val1 = val1 + val2;
            } else {
                val1 = Double.parseDouble(edtInput.getText().toString());
            }

            txtResult.setText(val1 + " + ");
            edtInput.setText(null);
        }
    });

    btnSub.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            ACTION = "-";
            if(!Double.isNaN(val1)){
                val2 = Double.parseDouble(edtInput.getText().toString());
                val1 = val1 - val2;
            } else {
                val1 = Double.parseDouble(edtInput.getText().toString());
            }

            txtResult.setText(val1 + " - ");
            edtInput.setText(null);
        }
    });

    btnMul.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            ACTION = "*";
            if(!Double.isNaN(val1)){
                val2 = Double.parseDouble(edtInput.getText().toString());
                val1 = val1 * val2;
            } else {
                val1 = Double.parseDouble(edtInput.getText().toString());
            }

            txtResult.setText(val1 + " * ");
```

```
            edtInput.setText(null);
        }
    });

    btnDiv.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            ACTION = "/";
            if(!Double.isNaN(val1)){
                val2 = Double.parseDouble(edtInput.getText().toString());
                val1 = val1/val2;
            } else {
                val1 = Double.parseDouble(edtInput.getText().toString());
            }

            txtResult.setText(val1 + " / ");
            edtInput.setText(null);
        }
    });

    btnResult.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            if(ACTION != null && ACTION.equals("+")){
                double result = val1 + Double.parseDouble(edtInput.getText().toString());
                txtResult.setText(null);
                edtInput.setText(String.valueOf(result));
            } else if(ACTION != null && ACTION.equals("-")){
                double result = val1 - Double.parseDouble(edtInput.getText().toString());
                txtResult.setText(null);
                edtInput.setText(String.valueOf(result));
            } else if(ACTION != null && ACTION.equals("*")){
                double result = val1 * Double.parseDouble(edtInput.getText().toString());
                txtResult.setText(null);
                edtInput.setText(String.valueOf(result));
            } else if(ACTION != null && ACTION.equals("/")){
                double result = val1 / Double.parseDouble(edtInput.getText().toString());
                txtResult.setText(null);
                edtInput.setText(String.valueOf(result));
            }

            ACTION = null;
            val1 = Double.NaN;
            edtInput.setSelection(edtInput.getText().length());
        }
    });

    btnNumber0.setOnClickListener(new View.OnClickListener() {
```

```
        @Override
        public void onClick(View v) {
            edtInput.setText(edtInput.getText() + "0");
            edtInput.setSelection(edtInput.getText().length());
        }
    });

    btnNumber1.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            edtInput.setText(edtInput.getText() + "1");
            edtInput.setSelection(edtInput.getText().length());
        }
    });

    btnNumber2.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            edtInput.setText(edtInput.getText() + "2");
            edtInput.setSelection(edtInput.getText().length());
        }
    });

    btnNumber3.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            edtInput.setText(edtInput.getText() + "3");
            edtInput.setSelection(edtInput.getText().length());
        }
    });

    btnNumber4.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            edtInput.setText(edtInput.getText() + "4");
            edtInput.setSelection(edtInput.getText().length());
        }
    });

    btnNumber5.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            edtInput.setText(edtInput.getText() + "5");
            edtInput.setSelection(edtInput.getText().length());
        }
    });

    btnNumber6.setOnClickListener(new View.OnClickListener() {
```

```
        @Override
        public void onClick(View v) {
            edtInput.setText(edtInput.getText() + "6");
            edtInput.setSelection(edtInput.getText().length());
        }
    });

    btnNumber7.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            edtInput.setText(edtInput.getText() + "7");
            edtInput.setSelection(edtInput.getText().length());
        }
    });

    btnNumber8.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            edtInput.setText(edtInput.getText() + "8");
            edtInput.setSelection(edtInput.getText().length());
        }
    });

    btnNumber9.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            edtInput.setText(edtInput.getText() + "9");
            edtInput.setSelection(edtInput.getText().length());
        }
    });

    btnDot.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            edtInput.setText(edtInput.getText() + ".");
            edtInput.setSelection(edtInput.getText().length());
        }
    });
  }
}
```

a) Explain working with different types of resources

- Animation Resources. Define pre-determined animations. *Tween animations* are saved in res/anim/ and accessed from the R.anim class. *Frame animations* are saved in res/drawable/ and accessed from the R.drawable class.
- Color State List Resource. Define a color resources that changes based on the View state. Saved in res/color/ and accessed from the R.color class.
- Drawable Resources. Define various graphics with bitmaps or XML. Saved in res/drawable/ and accessed from the R.drawable class.
- Layout Resource. Define the layout for your application UI. Saved in res/layout/ and accessed from the R.layout class.
- Menu Resource. Define the contents of your application menus. Saved in res/menu/ and accessed from the R.menu class.
- String Resources. Define strings, string arrays, and plurals (and include string formatting and styling). Saved in res/values/ and accessed from the R.string, R.array, and R.plurals classes.
- Style Resource. Define the look and format for UI elements. Saved in res/values/ and accessed from the R.style class.
- Font Resources. Define font families and include custom fonts in XML. Saved in res/font/ and accessed from the R.font class.
- More Resource Types. Define values such as booleans, integers, dimensions, colors, and other arrays. Saved in res/values/ but each accessed from unique R sub-classes (such as R.bool, R.integer, R.dimen, etc.).
- **String Resources**: A string resource provides text strings for your application with optional text styling and formatting. There are three types of resources that can provide your application with strings:

  String. XML resource that provides a single string.
  String Array. XML resource that provides an array of strings.
  Quantity Strings (Plurals). XML resource that carries different strings for pluralization.

Android supports several types of resources that can be used in an application. These resources include:

Layouts: Layouts define the UI of an application. They are used to specify the placement of UI elements on the screen. The most commonly used layouts are LinearLayout, RelativeLayout, FrameLayout, and ConstraintLayout.

Strings: Strings are used to store text that is displayed to the user. They can be stored in a strings.xml file and referenced using a resource identifier.

Colors: Colors can be used to specify the color of text, backgrounds, and other UI elements. They can be defined in a colors.xml file and referenced using a resource identifier.

Images: Images can be used to add visual elements to an application. They can be stored in the drawable folder and referenced using a resource identifier.

Dimensions: Dimensions are used to specify the size of UI elements. They can be defined in a dimens.xml file and referenced using a resource identifier.

Styles: Styles are used to define the appearance of UI elements. They can be defined in a styles.xml file and applied to UI elements using a style attribute.

b) Explain Web APIs, Network APIs, Telephony APIs,

Web APIs (Application Programming Interfaces) are a set of protocols, tools, and definitions that enable different software applications to communicate with each other over the internet. They provide a way for developers to access and manipulate data and functionality provided by remote servers, such as web services, databases, or other applications.

Web APIs typically use a standard set of request and response formats, such as JSON or XML, to facilitate communication between different applications. These APIs can be designed for a variety of purposes, including accessing data, performing calculations, or manipulating images and other media.

Web APIs can be classified into several categories, including RESTful APIs, SOAP APIs, and GraphQL APIs. RESTful APIs are the most common type and use HTTP methods such as GET, POST, PUT, and DELETE to access and manipulate resources.

Web APIs have become an essential component of modern web development, as they allow developers to easily integrate third-party services and data into their applications. Many popular websites and services, such as Facebook, Twitter, and Google, offer their own APIs to developers, enabling them to create new applications or extend the functionality of existing ones.

Network APIs (Application Programming Interfaces) are a set of protocols, tools, and definitions that enable software applications to communicate with each other over a computer network, such as the internet. They provide a way for developers to access and manipulate network resources, such as servers, routers, and switches, and to perform network operations, such as sending and receiving data packets.

Network APIs can be used for a variety of purposes, including network management, network monitoring, and network security. They typically use a standard set of protocols and data formats, such as TCP/IP, UDP, and SNMP, to facilitate communication between different applications.

Some examples of network APIs include:

- Socket API: It is a low-level network API that provides a way for applications to create and manage network sockets, which are the endpoints for sending and receiving data over a network.
- Network Information API: It provides information about the network connection, such as the type of network, IP address, and signal strength.
- Simple Network Management Protocol (SNMP) API: It allows developers to manage and monitor network devices, such as routers, switches, and servers, by querying and setting their configuration parameters.
- Network Security APIs: These APIs provide encryption and decryption of network traffic, authentication of users and devices, and other security-related functions.

Network APIs are an essential component of network programming and play a crucial role in the development of network-based applications and services. They enable developers to create powerful, scalable, and secure network applications that can communicate with other applications and services over the internet.

Telephony APIs (Application Programming Interfaces) are a set of protocols, tools, and definitions that enable software applications to interact with telephony services, such as voice calling, messaging, and video conferencing. They provide a way for developers to access and manipulate telephony resources, such as voice and SMS gateways, and to perform telephony operations, such as making and receiving phone calls, sending and receiving text messages, and initiating video calls.

Telephony APIs can be used for a variety of purposes, including building telephony-based applications, integrating telephony services into existing applications, and automating telephony tasks. They typically use a standard set of protocols and data formats, such as SIP, RTP, and JSON, to facilitate communication between different applications and services.

Some examples of telephony APIs include:

- Voice over IP (VoIP) API: It allows developers to create voice calling applications that use the internet instead of traditional telephone lines. It provides functions for making and receiving calls, call routing, call conferencing, and call recording.
- Short Message Service (SMS) API: It allows developers to send and receive text messages to and from mobile devices. It provides functions for sending and receiving messages, managing message queues, and monitoring message delivery status.
- Unified Communications (UC) API: It enables developers to create applications that integrate telephony services with other communication services, such as email, instant messaging, and video conferencing. It provides functions for initiating and managing communication sessions, managing contacts, and monitoring presence.

Telephony APIs are an essential component of telephony programming and play a crucial role in the development of telephony-based applications and services. They enable developers to create powerful, scalable, and secure telephony applications that can communicate with other applications and services over the internet.