ASSIGNMENT-1

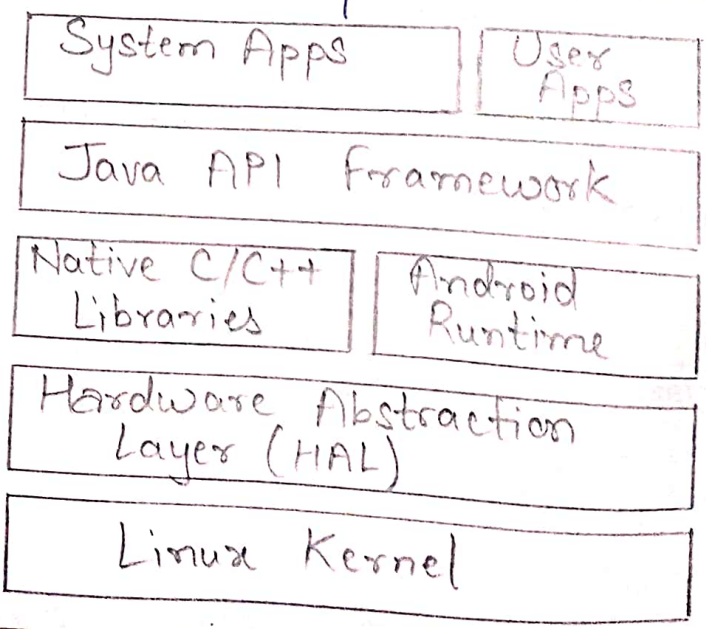
Q1. What is android? Why develop apps for android?

Ans: Android is an operating system and programming platform developed by Google for mobile phones and other mobile devices, such as tablets.

Developers create apps for a variety of reasons They may need to address business requirements or build new services or businesses, or they may want to offer games and other types of content for users. Developers choose to develop for Android in order to reach the majority of mobile device users.

Q2. With neat diagram, Show the major components of Android stack.

Ans: The following diagram shows the major components of the Android stack — the operating system and development architecture.



In the figure above:

1. Apps: Apps live at this level, along with c system apps for email, SMS, calendars e
2. Java API framework: All features for andr development, such as UI components, resou management and lifecycle management ar available through application programmir interfaces.
3. Libraries and Android runtime: Each app in its own process, with its own instan of the Android runtime.
4. Hardware Abstraction Layer: This layer pr standard interfaces that expose device ha capabilities to the higher-level Java API framework.
5. The Linux Kernel: The foundation of the a platform is the linux kernel. The layers o the linux kernel rely on the linux kern for threading, low-level memory manag and underlying functionality.

Q3. Explain the challenges faced in Andr app development.

Ans: While the android platform provides ri functionality for app development, there a still a number of challenges we need t address, such as:

- Building for a multiscreen world
- Getting performance right
- Keeping our code and our users more secure
- Making sure our app is compatible with older potform versions.
- Understanding the market and the user.

Q4. Explain the high-level picture of Android App development process.

Ans: 1. Create the project in Android Studio and choose an appropriate template.

2. Define a layout for each screen that has UI elements. We can place UI elements on the screen using the layout editor, or can write code directly in the Extensible Markup Language (XML).

3. Write code using the Java programming language.

4. Build and run the app on real and virtual devices.

5. Test and debug the app's logic and UI.

6. Publish the app by assembling the final APK (package file) and distributing it through channels such as Google play.

Q5. Explain the steps for creating a Virtual device?

Ans: To run an emulator on our computer, the AVD manager to create a configura that describes the virtual device.

Select Tools > Android > AVD Manager or cli the AVD Manager icon in the toolbar.

The Your Virtual Devices screen appears sh all of the virtual devices created previous Click the + Create Virtual Device button create a new virtual device.

We can select a device from a list of predefined hardware devices.

After we click Next, the System Image s appears for choosing the version of the An System for the device.

Click the Download link visible next to system image version and click Finish its done.

ASSIGNMENT-2

Q1. What do you mean by focus? Differentiate clickable and focusable attributes. Explain the algorithm used for Focus movement.

Ans: Focus indicates which view is selected. The user can initiate focus by tapping on a View, for example a specific EditText element.

- A focusable view is allowed to gain focus from a touchscreen, external keyboard, or other input device.
- A clickable view is any view that reacts to being tapped or clicked.

Focus movement is based on a natural algorithm that finds the nearest neighbour in a given direction:

- When the user taps the screen, the topmost View under the tap is in focus, providing touch access for the child View elements of the topmost View.
- If we set an EditText view to a single line, the user can tap the right-arrow key on the on-screen keyboard to close the keyboard and shift focus to the next input control view. Focus can also shift to a different View if the user interacts with a directional control, such as a D-pad or trackball.

Q2. How to explicitly set the focus?

Ans: We can explicitly set the focus or find o which view has focus by using the follo methods:

→ Use requestFocus to give focus to a specif View.

→ To change whether a View can take focu call setFocusable.

→ To set a listener that is notified when view gains or loses focus, use setOnFocusCh Listener.

Q3. How do you create a raised button with and text?

Ans: To create a button with text and an ic use a Button in your XML layout. Add the android:drawableLeft attribute to draw t icon to the left of the button's text, as in the figure below:

```
<Button
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:text = "@string/button_text"
    android:drawableLeft = "@drawable/button_i
/>
```

Q4. Explain the designing of the flat buttons.

Ans: A flat button, also known as a text button or boderless button, is a text-only button that looks flat and doesn't have a shadow. The major benefit of flat buttons is simplicity: a flat button doesn't distract the user from the main content as much as a raised button does. Flat buttons are useful for dialogs that require user interaction. In this case, we want the button to use the same font and style as the surrounding text to keep the look and feel consistent across all the elements in the dialog.

Q5. Explain Radio buttons designing along with onRadioButtonClicked() method.

Ans: We use radio buttons when we have two or more options that are mutually exclusive. When the user selects one, the others are automatically deselected.

We use the android:onClick attribute for each RadioButton to declare the click handler, which must be defined within the Activity that hosts the layout. In the layout above, clicking any RadioButton calls the same onRadioButtonClicked() method in the Activity. We could also create separate click handlers in the Activity for each RadioButton.

ASSIGNMENT-3

Q1. What is AsyncTask? With a neat diagram, explain the steps for execution of AsyncTask.

Ans. AsyncTask is a worker thread is any thread which is not the main or UI thread Use the AsynTask class to implement an asynchronous, long-running task on a worker thread. AsyncTask allows us to perform background operations on a worker thread and publish results on the UI thread without needing to directly manipulate threads or handlers.

When AsyncTask is executed, it goes through four steps:

1. onPreExecute() is invoked on the UI thread before the task is executed.
2. doInBackground(Params...) is invoked on the background thread immediately after onPreExecute() finishes.
3. onProgressUpdate(Progress...) runs on the UI thread after publishProgress(Progress...) is invoked Use onProgressUpdate() to report any form of progress to the UI thread while the background computation is executing.
4. onPostExecute(Result) runs on the UI thread after the background computation has finished.

Q2. List out the limitations of AsyncTask.

Ans: AsyncTask is impractical for some use case

- Changes to device configuration cause probl
When device configuration changes while an AsyncTask is running, for example if the u changes the screen orientation, the activity that created the AsyncTask is destroyed a re-created. The AsyncTask is unable to acc the newly created activity, and the resu of the AsyncTask aren't published.
- Old AsyncTask objects stay around, and your a may run out of memory or crash.
If the activity that created the AsyncTask destroyed, the AsyncTask is not destroyed al with it.

Q3. Define Loader? Explain, how do you start restart a loader?

Ans: Background tasks are commonly used to data such as forecast reports or movie re Loading data can be memory intensive, an want the data to be available even if th device configuration changes. For these situat use loaders which are classes that facilita loading data into an activity.

→ Starting a Loader:

Use the LoaderManager class to manage one

more loader instances within an activity or fragment. We use initLoader() to initialize a loader and make it active. Typically, we do this within the activity's onCreate() method.

→ Restarting a loader:

When initLoader() reuses an existing loader, it doesn't replace the data that the loader contains but sometimes we want it to. For example, when we use a user query to perform a search and the user enters a new query, we want to reload the data using the new search term.

Q4. What is AsyncTaskLoader? How do you use AsyncTaskLoader?

Ans: AsyncTaskLoader is the loader equivalent of AsyncTask. AsyncTaskLoader provides a method, loadInBackground(), that runs on a separate thread. The results of loadInBackground() are automatically delivered to the UI thread, by way of the onLoadFinished() LoadManager callback.

AsyncTaskLoader Usage:

To define a subclass of AsyncTaskLoader, create a class that extends AsyncTaskLoader<D>, where D is the data type of the data you are loading

Q5. List and explain the classes for managing the network state?

Ans: To check the network connection, use the following classes:

- ConnectivityManager answers queries about state of network connectivity. It also notifies apps when network connectivity changes.
- NetworkInfo describes the status of a network interface of a given type.

The following code snippet tests whether Wi-Fi and mobile are connected. In the code:

- The getSystemService() method gets an instance of ConnectivityManager from the context.
- The getNetworkInfo() method gets the status of the device's Wi-Fi connection, then its mobile connection. The getNetworkInfo() method returns a NetworkInfo Object, which contains information about the given network's connection status.
- The networkInfo.isConnected() method returns true if the given network is connected. If a network is connected, it can be used to establish sockets and pass data.

ASSIGNMENT-4

Q1. Define shared preferences. List and explain data storage options available.

Ans: Shared preferences allow us to store small amounts of primitive data as key-value pairs in a file on the device.

Data Storage options include the following:

→ Shared preferences: Store private primitive data in key-value pairs.

→ Internal Storage: Store private data on the device memory.

→ External storage: Store public data on the shared external storage.

→ SQLite databases: Store structured data in a private database.

→ Room persistence library: Part of the Android Architecture Component libraries. Room caches an SQLite database locally, and automatically syncs changes to a network database.

→ Cloud backup: Back up your app and user data in the cloud.

→ Firebase realtime database: Store and sync data with a NoSQL cloud database.

→ Custom data store: Configure the Preference APIs to store preferences in a storage location you provide.

Q2. Differentiate between internal storage and external storage.

Ans:

Internal Storage	External Storage
Always available.	Not always available, because the user can mount the external storage as USB storage.
Only our app can access files.	World-readable. Any app can read.
When the user uninstalls your app, the system removes all your app's files from internal storage.	When the user uninstalls your app, the system removes your app's file from here only if we save them in the directory from getExternalFilesDir().
Internal storage is best when we want to be sure that neither the user nor other apps can access our files.	External storage is the best place for files that don't require access restrictions and for files that we want to share with other apps or allow the user to access with a computer.

3. Differentiate between Shared preferences and saved instance state.

ns:

Shared Preferences	Saved Instance state
Persists across user sessions, even if oour app is stopped and restarted, or if the device is rebooted.	Preserves state data across activity instances in the same user session.
Used for data that should be remembered across user sessions, such as a user's preferred settings or their game score.	Used for data that should b not be remembered across sessions, such as the currently selected tab, or any current state of an activity.
Represented by a small number of key-value pairs.	Represented by a small number of key-value pairs.
Data is private to the app.	Data is private to the app.
Common use is to store user preferences.	Common-use is to recreate state after the device has been rotated.

Q4. How do you create, save and restore the Shared preferences files? Explain.

Ans: We need only one shared preferences fi for our app, and it is customarily nam with the package name of our app. This m its name unique and easily associated wit our app.

We create the shared preferences file in the onCreate() method of our main activity a store it in a member variable.

We save preferences in the onPause() state the activity lifecycle using the SharedPrefere Editor interface.

We restore shared preferences in the onCreate() method of our activity. The "gets" methods su as getInt() or getString() take two argument one for the key and one for the default va if the key cannot be found. Using the defau argument, we don't have to test whether the preference exists in the file.

Q5. Explain the mechanism for listening share preferences.

Ans: There are several reasons we might want to be notified as soon as the user changes one of the preferences. In order to receive callback when a change happens to any one of the preferences, implement the Shared

ference.OnSharedPreferenceChangeListener interface and register the listener for the SharedPreferences object by calling registerOnSharedPreferenceChangeListener().

The interface has only one callback method, onSharedPreferenceChanged(), and we can implement the interface as a part of our activity.

ASSIGNMENT-5

1. Define Permissions. Explain how to request permissions. How do you grant and revoke permissions?

Ans: Strating from Android 6.0 (API 23), users are not asked for permissions at the time of installation rather developers need to request the permissions at the run time. Only the permissions that are defined in the manifest file can be requested at run time.

Steps for requesting permission:

Step 1: Declare the permission in the Android Manifestfile: In Android, permissions are declared in the Android Manifest.xml file using the uses-permission tag.

Step 2: Modify activity_main.xml file to Add two buttons to request permission on button click

Step 3: Check whether permission is already granted or not. If permission isn't already granted, request the user for the permission.

The user has the right to revoke permissions from any app at any time, even if the app targets a lower API level.

Q2. Define Firebase.

Ans. Firebase is a mobile platform that helps us develop apps, grow our user base and earn more money. Firebase is made up of complementary features that you can mix-and-match to fit your needs.

Some features are Analytics, Cloud Messaging, Notifications, and the Test Lab.

For data management, Firebase offers a Realtime Database.

→ Store and sync data with a NoSQL cloud database.
→ Connected apps share data.
→ Hosted in the cloud.
→ Data is stored as JSON.
→ Data is synchronized in real time to every connected client
→ Data remains available when our app goes offline.

Q3. Explain Firebase Analytics.

Ans. Analytics are at the heart of successful apps so when we expand. Firebase Analytics is designed to make our analytics data actionable. Google Analytics for firebase provides free, unlimited reporting on up to 500 distinct

events. The SDK automatically captures certain key events and user properties and we can define our own custom events to measure the things that uniquely matter to our business.

Q4. What is Firebase Realtime database ?

Ans: The Firebase Realtime Database is a cloud hosted database. Data is stored as JSON and synchronized in realtime to every connected client. When we build cross-platform apps with our Apple platforms, Android, and JavaScript SDKs, all of our clients share one Realtime Database instance and automatically receive updates with the newest data.