Model Question Paper-1

Mobile Application Development

2 Marks:

1) What is Web App and mobile App?

A) Web app: A web app is an application that runs on a web server and is accessed through a web browser over the internet. Web apps are designed to be accessible from any device with a browser, whether it's a desktop, laptop, tablet, or smartphone.

Ex: Canva, Google, Google Docs

Mobile App: A mobile application (also called a mobile app) is a type of application designed to run on a mobile device, which can be a Smartphone or tablet. Even if apps are usually small software units with limited function, they still manage to provide users with quality services and experiences.

Ex: WhatsApp, Uber, Spotify, Instagram.

2) Define activity?

A) An activity is a class that represents a single screen in android. It is like window or frame of Java. By the help of activity, you can place all your UI components (button, label, text field etc.) or widgets in a single screen. It is the main entry point for user interaction. Any application, don't matter how small it is (in terms of code and scalability), has at least one Activity class. You can have multiple activities in your app.

Ex: Layout File (activity_main.xml): Defines the UI elements.
Java Class (MainActivity.java): Handles the logic and functionality.

3. What is contextual action bar?

A) A **Contextual Action Bar (CAB)** in Android is a temporary action bar that appears in the app's UI in response to a user selecting an item (or items). It provides actions that are contextually related to the selected items. Unlike the standard action bar, which is always present, the contextual action bar only appears when needed, typically to offer actions that can be performed on the selected items.

4. What is Content provider?

A) A content provider component supplies data from one application to others on request. Such requests are handled by the methods of the ContentReslover class. It's a way to share data between different applications securely. A content provider can use different ways to store its data and data can be stored in a database, in files, or even over a network.

5. **What is Android Studio?**

A) **Android Studio** is the official Integrated Development Environment (IDE) for Android app development, based on JetBrains' IntelliJ IDEA software. It includes a code editor for writing app code, a visual interface builder for designing app screens, and tools to test and debug apps on virtual or real Android devices. It is the main tool used by developers to make Android apps efficiently.

6. **What are types of layouts in UI?**

Linear Layout:

Arranges elements in a single line, either horizontally or vertically.

Relative Layout:

Positions elements relative to each other or the parent container.

Constraint Layout:

Positions elements relative to each other and the parent using constraints.

Frame Layout:

Displays a single item at a time, typically used for holding a single view or fragment.

Table Layout:

Organizes elements in rows and columns, similar to a table structure.

4 marks:

7) Discuss option menu and context menu.

Menus are useful for displaying additional options that are not directly visible on the main UI of an application. There are two types of menus in application.

a) Option Menu
b) Context menu

Option menu: Option menu is a primary collection of menu items for an activity and it is useful to implement actions that have a global impact on the app, such as settings, search etc. It appears when the user presses the menu button on their device or the three-dot overflow button in the app's action bar.

- Used for global actions within the application.
- Typically accessed via the action bar or overflow menu.
- Common actions include settings, search, and other app-wide actions.

Code:

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:id="@+id/action_search"
        android:title="Search"
        android:icon="@drawable/ic_search"
        android:showAsAction="ifRoom"/>
    <item
        android:id="@+id/action_settings"
        android:title="Settings"
        android:showAsAction="never"/>
</menu>
```

Context Menu: The context menu is a floating menu that appears when the user performs a long-click (tap and hold) on a view. It's typically used for actions that affect the selected item or context in which the menu was activated.

- Used for context-specific actions related to a particular view or item.
- Accessed via a long-click on a view.
- Common actions include edit, delete, and other item-specific actions.

8. Explain Key mobile application services?

Key Mobile Application Services:

Certainly! When it comes to **mobile application services**, there are several crucial aspects to consider. Let's explore some of the key services:

1. **User Sign-up/Sign-in and Management**:
    - This service involves creating a seamless experience for users to register, sign in, and manage their accounts within the mobile app. It includes features like email-based registration, social login (such as
    - Face book or Twitter), and password recovery.
2. **Social Login**:
    - Social login allows users to sign in to your app using their existing social media credentials (e.g., Face book, Google, Twitter). It simplifies the authentication process and enhances user convenience.
3. **Analytics and User Engagement**:
    - Analytics services help track user behavior within the app. By analyzing data such as user interactions, session duration, and conversion rates, you can make informed decisions to improve the app's performance and engagement.
4. **Push Notifications**:
    - Push notifications keep users informed and engaged by sending timely updates, reminders, or personalized messages directly to their devices. Effective push notification strategies can enhance user retention and drive app usage.
5. **Real Device Testing**:
    - Ensuring your app works flawlessly across various devices and operating systems is essential. Real device testing involves testing your app on actual devices (not just simulators) to identify any issues related to performance, compatibility, or usability.

Remember that these services play a critical role in delivering a successful mobile app experience. Whether you're developing for iOS, Android, or cross-platform, thoughtful implementation of these services contributes to user satisfaction and app success.

9) **What is JSON and SDK?**

JSON (JavaScript Object Notation):

JSON remains a fundamental format for exchanging data between servers and Android applications. It's lightweight, easy to read, and easy to parse. Android apps often use JSON to fetch data from web services, APIs, or local files. Here's a basic example of JSON data representing a user:

```
{
  "name": "John Doe",
  "age": 30,
  "city": "New York",
  "interests": ["Reading", "Coding", "Traveling"]
}
```

Android provides libraries like JSONObject and JSONArray to parse JSON data received from a server or stored locally.

SDK (Software Development Kit):

The Android SDK (software development kit) is a set of development tools used to develop applications for Android platform. SDK provides a selection of tools required to build Android apps or to ensure the process goes as smoothly as possible. In Android development, the **Android SDK** is a comprehensive set of tools, libraries, and documentation provided by Google. It includes:

- **Android Studio**: The official Integrated Development Environment (IDE) for Android development.
- **Android API libraries**: Libraries that provide access to various Android platform features like UI components, sensors, multimedia, networking, and more.
- **Emulator**: A virtual device that allows developers to test their apps on different Android versions and device configurations.
- **Documentation**: Extensive guides, reference materials, and tutorials to help developers build Android applications effectively.
- **Support Libraries**: Additional libraries like AndroidX to ensure backward compatibility and support new features on older Android versions.

Developers use the Android SDK along with Java or Kotlin programming languages to create Android apps that run on smartphones, tablets, wearables, TVs, and other Android-powered devices. It provides all the necessary resources to develop, debug, test, and deploy Android applications efficiently.

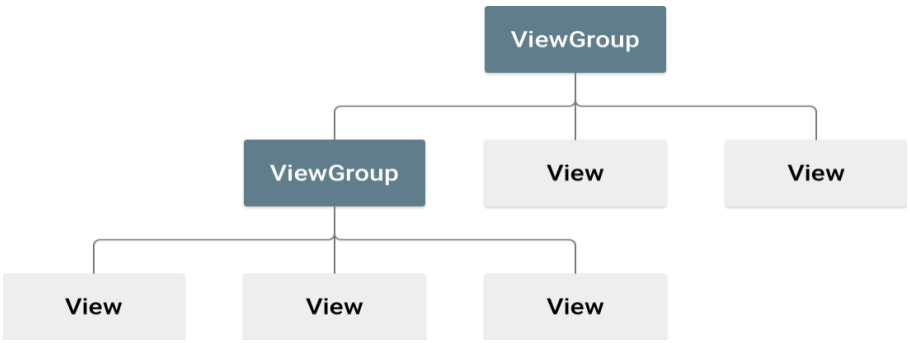
10) Explain the features of Android OS.

A) The important features of android are given below:

1. It is open-source.
2. Anyone can customize the Android Platform.
3. There are a lot of mobile applications that can be chosen by the consumer.
4. It provides many interesting features like weather details, opening screen, live RSS (Really Simple Syndication) feeds etc.
5. It provides support for messaging services (SMS and MMS), web browser, storage (SQLite), connectivity (GSM, CDMA, Blue Tooth, and Wi-Fi etc.), media, handset layout etc.
6. Storage — Uses SQLite, a lightweight relational database, for data storage.
7. Connectivity — Android devices support a wide range of connectivity options including GSM, EDGE, IDEN, CDMA, EV-DO, UMTS, Bluetooth, Wi-Fi, LTE, and WiMAX.
8. Messaging — Supports both SMS and MMS.
9. Web browser — based on the open source WebKit, together with Chrome's V8 JavaScript engine.
10. Media support — Android offers comprehensive media support including formats like H.263, H.264, MPEG-4, AMR, AAC, MP3, MIDI, Ogg Vorbis, WAV, and various image formats (JPEG, PNG, GIF, BMP).
11. Hardware support — Accelerometer Sensor, Camera, Digital Compass, Proximity Sensor, and GPS
12. Multi-touch — Supports multi-touch screens
13. Multi-tasking — Supports multi-tasking applications
14. Flash support — Android 2.3 supports Flash 10.1.
15. Tethering — supports sharing of Internet connections as a wired/wireless hotspot.

11) Explain View group.

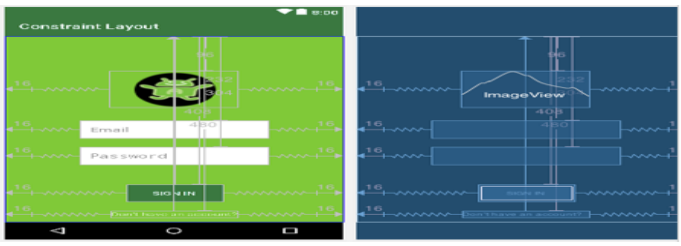
- An activity contains *views* and *ViewGroups*. A View usually draws something the user can see and interact with. A view is a widget that has an appearance on screen. Examples of views are buttons, labels, and text boxes.
- A view derives from the base class **android.view.View.**
- In ViewGroup, one or more Views can be grouped together. A ViewGroup provides the layout, in which you can set the order of the appearance and sequence of the Views. Some examples of ViewGroups are LinearLayout and FrameLayout, AbsoulteLayout, Constraint Layout.
- It is derived from the base class **android.view.ViewGroup**.



Types of ViewGroups:

ConstriantLayout:

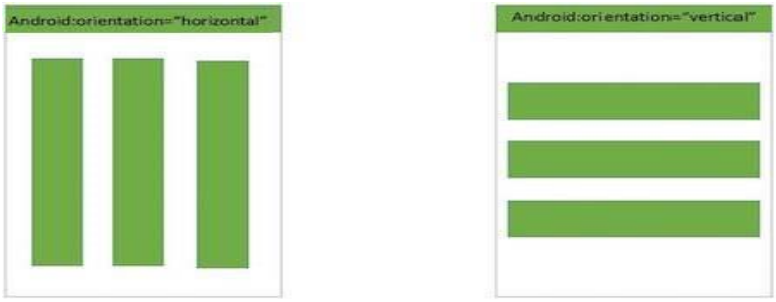
- ConstraintLayout is a ViewGroup subclass; whenever you open the android studio framework the application will be present in the constraint layout. In this we have to set the constraint in all four sides.
- This is the default layout
- Ex: Designing a login screen for a mobile app using constraint layout in Android.



For example, you can specify that the logo is centered horizontally in the parent, the username input field is centered below the logo, the password input field is centered below the username field, and so on.

LinearLayout:

- The LinearLayout arranges views in a single column or a single row. Child views can be arranged either horizontally or vertically in a single direction, which explains the need for two different layouts.



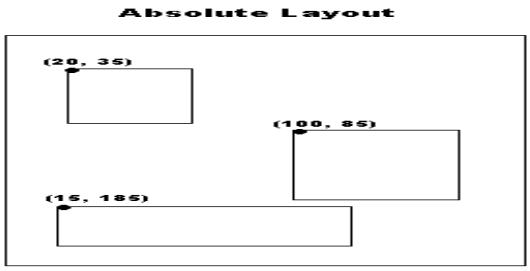
- The root layout is a LinearLayout with **android:orientation="horizontal"**, which means child views, will be placed side by side. Suitable for toolbars, menus, and other horizontally aligned components.

- The root layout is a Linear Layout with **android:orientation="vertical",** which means child views will be placed one below the other. Suitable for forms, lists, and other vertically aligned components.

AbsoluteLayout:

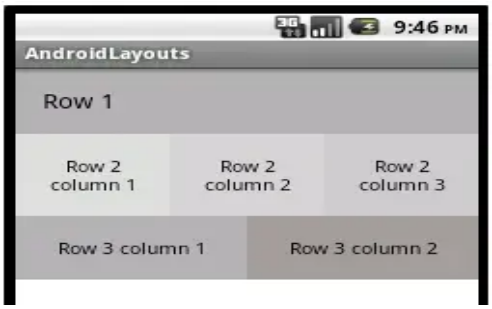
- An Absolute layout allows you to specify the exact location. i.e., X and Y coordinate of its children with respect to the origin at the top left corner of the layout.
- android:layout_x
  - This specifies the x-coordinate of the viev
- android:layout_y
  - This specifies the y-coordinate of the viev

android:layout_x="50px"
android:layout_y="361px"



TableLayout:

- TableLayout is a view that groups views into rows and columns. You use the <TableRow> element to designate a row in the table. Each row can contain one or more views.



RelativeLayout:

- RelativeLayout is a view group that displays child views in relative positions. The position of each view can be specified as relative to sibling elements or relative to the parent.
- These attributes are as follows:
  - **layout_alignParentTop:** Aligns the top edge of the view with the top edge of the parent.
  - **layout_alignParentStart:** Aligns the start edge of the view with the start edge of the parent.
  - **layout_alignStart:** Aligns the start edge of the view with the start edge of another specified view.

- **layout_alignEnd:** Aligns the end edge of the view with the end edge of another specified view.
- **layout_below:** Positions the view directly below another specified view.
- **layout_centerHorizontal:** Centers the view horizontally within the parent.

FrameLayout:

- The FrameLayout is a placeholder on screen that you can use to display a single view. Views that you add to a FrameLayout are always anchored to the top left of the layout.
- You can add multiple views to a FrameLayout, but each is stacked on top of the previous one. This is when you want to animate a series of images, with only one visible at a time.

Example: Image Carousel with FrameLayout
Imagine you want to create an image carousel where multiple images are displayed one after another with a fade-in animation, while a caption is shown on top of each image.

ScrollView

- A ScrollView is a special type of FrameLayout in that it enables users to scroll through a list of views that occupy more space than the physical display.
- Android supports vertical scroll view as default scroll view. Vertical scrollview scrolls element s vertically. Android uses Horizontal ScrollView for scrolls element horizontally.

GridView

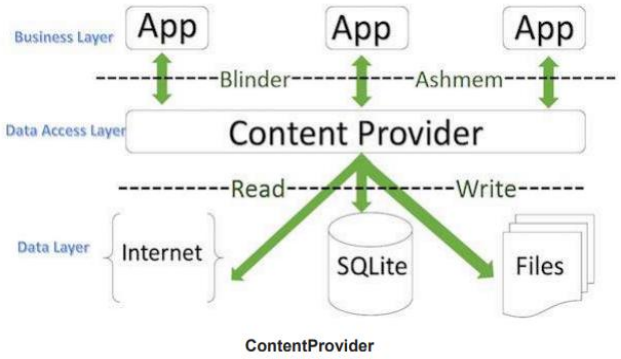
- GridView is a ViewGroup that displays items in a two-dimensional, scrollable grid.

ListView

- ListView is a view group that displays a list of scrollable items.

12. Explain content provider.

A) A content provider component supplies data from one application to others on request. Such requests are handled by the methods of the ContentReslover class. It's a way to share data between different applications securely. A content provider can use different ways to store its data and data can be stored in a database, in files, or even over a network.



Here's a breakdown of the storage options:

Database: The content provider can store its data in a database, typically a SQLite database. This is a common method because databases are efficient for handling structured data and complex queries.

Files: Data can also be stored in files. This method might be used for handling unstructured data, such as media files (images, videos) or large datasets.

Network: A content provider can also store data over a network. This could mean the data is fetched from a remote server or cloud storage.

Android ships with many useful content providers, including the following:

➢ Browser—Stores data such as browser bookmarks, browser history, and so on

➢ CallLog—Stores data such as missed calls, call details, and so on

➢ Contacts—Stores contact details

Content URI is the key concept of Content Providers. To access the data from a content provider, Content URI is used as a query string.

Structure of a Content URI is **Content: //authority /optionalPath /optionalID**

Content: It represents that the given URI is content URI.

Authority: Name of the Content provider like contacts, browser, and etc.It must be unique for every content provider.

OptionalPath: Specifies the type of data provided by the Content provider. For example, if you are getting all the contacts from the **Contacts** content provider, then the data path would be people and URI would look like this **content://contacts/people**

OptionalID: It is a numeric Value it specifies the specific record requested. For example, example, if you are looking for a contact number 5 in the Contacts content provider then URI would look like this **content: //contacts/people/5.**

Operations in Content provider: The fundamental operations are possible in content Provider namely Create, Read, update, get type, and delete. These operations are often termed as CRUD operations.

1. **OnCreate ( ):** This method is called when the provider is started.
2. **Query ( ):** This method receives a request from a client. The result is returned as a Cursor object.
3. **insert( ):** This method inserts a new record into the content provider.
4. **delete( ):** This method deletes an existing record from the content provider.
5. **update( ):** This method updates an existing record from the content provider.
6. **getType( ):** This method returns the MIME type of the data at the given URI.

8 Marks:

13. Explain the types of Intents.

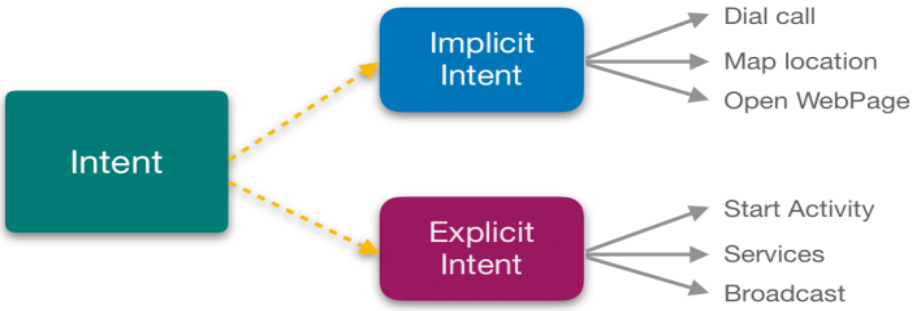
A) In Android, you navigate between activities through what is known as intent.

- Intent is a messaging object used to request any action from another app component. Intent's most common use is to launch a new activity from the current activity.
- Intent facilitates communication between different components. Intent object is used to call other activities.

The intent is used to launch an activity, start the services, broadcast receivers, display a web page, dial a phone call, send messages from one activity to another activity, and so on.

TYPES OF INTENTS:

- Intents are of two types:



- Explicit Intent is used to invoke a specific target component. It is used to switch from one activity to another activity in the same application. It is also used to pass data by invoking the external class.

- Explicit Intents specify the target component by providing the exact class name of the component to be invoked.

Example:-

1. We can use explicit intent to start a new activity when the user invokes an action or plays music in the background or on click button go to another activity.

   Intent intent = new Intent(MainActivity.this, SecondActivity.class);
   startActivity(intent);

   - MainActivity.this specifies the current context, which is the MainActivity.
   - SecondActivity.class specifies the target activity to which you want to navigate.
   - **For Example:** If we know class name then we can navigate the app from One Activity to another activity using Intent.

2. In amazon app if you go to Home page you can see prime, fresh, mobiles, electronics etc. If you click prime it transit to prime page activity likewise other tab also works.

IMPLICIT INTENT:

- An implicit intent is used when you want to perform an action, but you don't care which component performs it.
- The system will determine the appropriate component to handle the intent based on the available components that can respond to it.
- Example: Suppose you want to open a website URL in a web browser. You don't know which browser the user prefers, so you'll use an implicit intent:

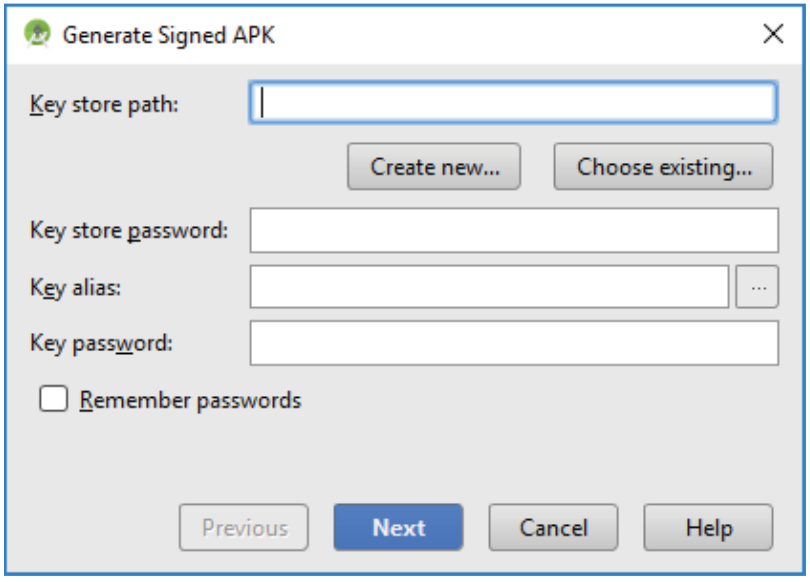
```
//Intent object and open the webpage
Intent intent = new Intent(Intent.ACTION_VIEW,Uri.parse("https://example.com"));
 startActivity(intent); //call a webpage
```

- Intent.ACTION_VIEW is the action you want to perform, which is viewing content.
- Uri.parse("https://example.com") specifies the data you want to view, which is a website URL.
  - Explicit intents are used for navigating within your own by specifying the target component, while implicit intents are used for performing actions where the system determines the appropriate component to handle the request.

14. Explain the steps of generating apk file.

Steps to Generate an APK File

1. **Prepare Your Project**:
   - **Ensure Code Completeness**: Make sure your project is complete, with all the necessary code, resources, and dependencies.
   - **Remove Debugging Code**: Clean up any debug code or logs that should not be included in the release version.
   - **Update Version Information**: Update the version number and version code in the build.gradle file to reflect the new release.
2. **Build and Sign the APK**:
   - Open your project in Android Studio.
   - Navigate to the **Build** menu at the top of the screen.
   - Select **Generate Signed Bundle / APK...**.
3. **Select APK**:
   - In the dialog that appears, choose **APK** and click **Next**.
4. **Create or Select a Key Store**:
   - **Key Store Location**: Specify the location where your keystore file will be saved. If you don't have a keystore, you can create a new one here.
   - **Key Store Password**: Enter the password for your keystore.
   - **Key Alias**: Create or select a key alias. This is a name that identifies the key within the keystore.
   - **Key Password**: Enter the password for the key.
   - Click **Next** after filling out the necessary information.



5. **Specify Build Variants**:

- **Build Type**: Select the build type, usually release for the final production version.
- **Flavors**: If your project uses product flavors, select the appropriate flavor.
- Click **Finish** to start the build process.

6. **Build Process**:
   - Android Studio will start the build process. This may take a few minutes, depending on the size of your project and the performance of your development machine.
   - You can monitor the progress in the **Build** window at the bottom of Android Studio.
7. **Locate the Generated APK**:
   - Once the build is complete, a dialog will appear showing the location of the generated APK file(s).
   - The APK files are typically located in the app/build/outputs/apk/ directory of your project.
8. **Test the APK**:
   - Before distributing the APK, thoroughly test it on various devices and configurations to ensure it works correctly.
   - Install the APK on a physical device or emulator by transferring the APK file to the device and running it.
9. **Distribute the APK**:
   - **Google Play Store**: To distribute via the Google Play Store, upload the signed APK file to the Google Play Console, complete the necessary details (app description, screenshots, etc.), and publish the app.
   - **Other Methods**: You can also distribute the APK directly to users through your website, email, or other distribution channels.

15) Explain the role of GPS?

- **Global Positioning System (GPS)** is a satellite-based navigation system that provides location and time information to a GPS receiver anywhere on or near the Earth where there is an unobstructed line of sight to four or more GPS satellites.
- The role of GPS in various applications, including Android devices, is crucial for providing accurate and reliable location-based services.

Key Functions of GPS:

1. **Positioning**:
   - **Determining Location**: GPS calculates the exact geographic location (latitude, longitude, and altitude) of a GPS receiver. This is achieved by triangulating signals from at least four GPS satellites.
   - **Navigation**: Enables users to navigate from one location to another by providing real-time location updates and directions.
2. **Timing**:

- **Time Synchronization**: GPS provides highly accurate time synchronization, which is critical for various applications such as telecommunications, financial networks, and power grids.

3. **Tracking**:
   - **Real-Time Tracking**: Used for tracking the movement of objects, vehicles, and people in real-time. This is commonly used in fleet management, logistics, and personal tracking applications.
4. **Mapping and Geographical Information Systems (GIS)**:
   - **Map Creation and Updates**: Helps in creating and updating digital maps by providing precise location data.
   - **GIS Integration**: Enhances GIS applications by providing accurate geographical data for analysis and decision-making.

Role of GPS in Android Devices:

1. **Location-Based Services**:
   - **Google Maps and Navigation**: GPS enables apps like Google Maps to provide turn-by-turn navigation, traffic updates, and points of interest.
   - **Local Search**: Helps in finding nearby businesses, restaurants, and services based on the user's current location.
2. **Geotagging**:
   - **Photos and Social Media**: Allows users to tag their photos and social media posts with their current location, enhancing the context of shared content.
3. **Fitness and Health Apps**:
   - **Activity Tracking**: Fitness apps use GPS to track outdoor activities like running, cycling, and hiking by recording routes, distances, and speeds.
4. **Safety and Emergency Services**:
   - **Emergency Response**: In case of emergencies, GPS can provide accurate location information to emergency services, ensuring a quicker and more effective response.
   - **Child and Pet Safety**: GPS-enabled devices help in tracking the location of children and pets, providing peace of mind to guardians.
5. **Augmented Reality (AR)**:
   - **Enhanced Experiences**: GPS plays a crucial role in AR applications by providing location context, which enhances the user experience by overlaying digital information on the real world.

16) Create an application to send Email?

Step By Step sending an implementation of sending a email:

Step: 1 Project Setup

Create an android project by selecting “Empty Views activity” and naming the project “email-example” and Choosing java as a language.

Step: 2 Create the UI

Design the UI to input the recipients email address, subject, and message body. This UI will allow users to enter the necessary details to send an email.

activity main.xml code:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical"
android:padding="16dp">

<EditText
android:id="@+id/email_address"
android:layout width="match_parent"
android:layout_height="wrap_content"
android:hint="Email Address"
android: inputType="textEmailAddress" />

<EditText
android:id="@+id/email_subject"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:hint="Subject"
android: inputType="text" />

<EditText
android:id="@+id/email_message"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:hint="Message"
android: inputType="textMultiline" />

<Button
android:id="@+id/send_email"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:text="Send Email" />

</LinearLayout>
```

Explanation:

* The LinearLayout' provides a vertical layout for the Ul components.
* The 'EditText' fields allow the user to input the recipient's email address, subject, and message content.
* The "Button' triggers the email-sending functionality when clicked.

Step3. Implement Main Activity

Implement the activity to handle sending emails using intents. This activity manages the user Interactions and handles the email-sending process.

```
public class MainActivity extends AppCompatActivity (
private EditText emailAddress;
private EditText emailsubject;
private EditText emailMessage;
private Button sendEmail;
@Override
protected void onCreate(Bundle savedInstanceState)
 super.onCreate(savedInstanceState);
 setContentView(R.layout.activity_main);
 emailAddress = findViewById(R.id.email_address);
 emailSubject = findViewById(R.id.email_subject);
 emailMessage = findViewById(R.id.email_message);
 sendEmail = findViewById(R.id.send_email);
 sendEmailButton.setOnClickListener(new View.OnClickListener() {
 @Override
  public void onClick(View v) {
    sendEmail();

                }
      });
}

private void sendEmail() {

String emailAddress = emailAddress.getText().toString();
String subject = emailSubject.getText().toString();
String message = emailMessageEditText.getText().toString();

Intent emailIntent = new Intent(Intent.ACTION_SEND);
emailIntent.setType("message/rfc822");
emailIntent.putExtra (Intent.EXTRA_EMAIL, new String[] {emailAddress});
emailIntent.putExtra(Intent.EXTRA_SUBJECT, subject);
emailIntent.putExtra (Intent.EXTRA_TEXT, message);
if (emailIntent.resolveActivity(getPackageManager()) != null) {
      startActivity(emailIntent);
```

```
    }
    else {
        Toast.makeText (this, "Email Client Not Installed", Toast.LENGTH_SHORT).show ();
        }
}
```

14. Explain the steps to create, open and close the database cursors.

Creating, opening, and closing database cursors are fundamental tasks when working with SQLite databases in Android. Here's a step-by-step explanation of these processes:

Steps to Create, Open, and Close Database Cursors in Android

1. **Create a Database Helper Class**:
    - The first step is to create a class that extends SQLiteOpenHelper. This class will manage database creation and version management.

```
public class DatabaseHelper extends SQLiteOpenHelper {

    private static final String DATABASE_NAME = "example.db";
    private static final int DATABASE_VERSION = 1;

    public DatabaseHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        // Create tables
        db.execSQL("CREATE TABLE example_table (id INTEGER PRIMARY KEY,
name TEXT, age INTEGER)");
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        // Handle database upgrade
        db.execSQL("DROP TABLE IF EXISTS example_table");
        onCreate(db);
    }
}
```

Open the Database:

- To interact with the database, you need to get a readable or writable database instance using the helper class.
  DatabaseHelper dbHelper = new DatabaseHelper(context); SQLiteDatabase db = dbHelper.getWritableDatabase(); // For read and write operations.

Create a Cursor:

- A Cursor object is used to perform database queries and retrieve results. You can create a cursor using query() or rawQuery() methods.

```
// Using query() method
Cursor cursor = db.query(
    "example_table",   // The table to query
    new String[]{"id", "name", "age"},  // The columns to return
    null, // The columns for the WHERE clause
    null, // The values for the WHERE clause
    null, // Don't group the rows
    null, // Don't filter by row groups
    null  // The sort order
);

// Using rawQuery() method
Cursor cursor = db.rawQuery("SELECT id, name, age FROM example_table", null);
```

Use the Cursor:

- Move the cursor to the first row and iterate through the results to read data.

```
if (cursor.moveToFirst()) {
    do {
        int id = cursor.getInt(cursor.getColumnIndexOrThrow("id"));
        String name = cursor.getString(cursor.getColumnIndexOrThrow("name"));
        int age = cursor.getInt(cursor.getColumnIndexOrThrow("age"));
        // Process the data
    } while (cursor.moveToNext());
}
```

Close the Cursor:

- After you are done using the cursor, it is important to close it to release resources.

```
cursor.close();
```

Close the Database:
Finally, close the database to release resources.dbHelper.close ();

- **Create**: Use a helper class extending SQLiteOpenHelper to create and manage the database.
- **Open**: Obtain a readable or writable instance of the database using the helper class.
- **Create Cursor**: Execute queries to create a Cursor object for retrieving data.
- **Use Cursor**: Iterate over the cursor to process the query results.
- **Close Cursor**: Always close the cursor after use to free up resources.
- **Close Database**: Close the database when done to free up resources.

18. Explain the procedure to publish android app?

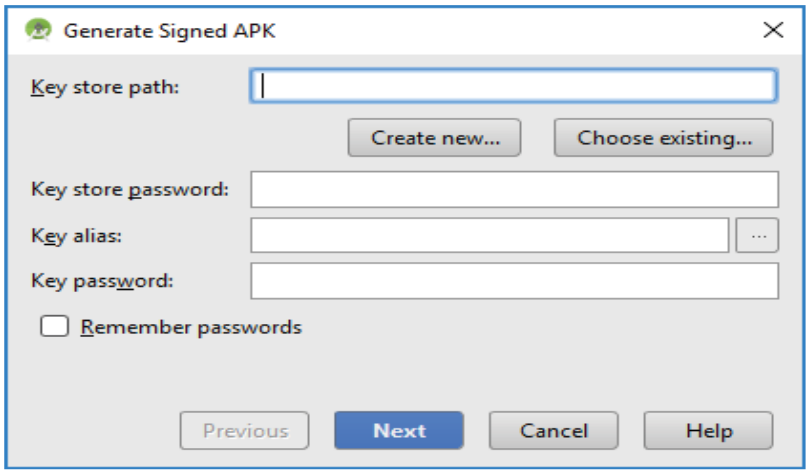
A) Publishing Your Application

- After you have created, and fully debugged, your application, you might want to deploy it to the GoogleStore for others to enjoy. The following sections outline the steps for publishing your applications.

Generating a Signed APK

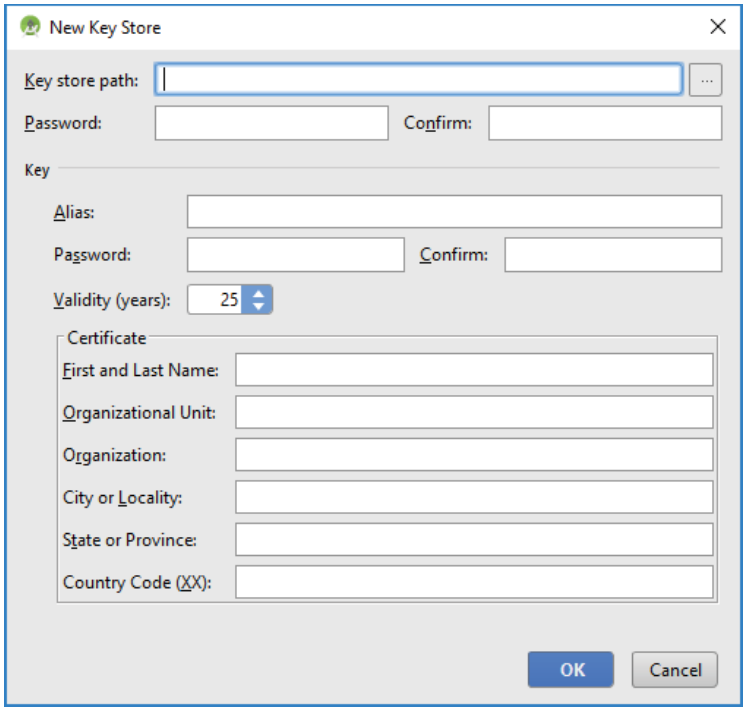
- To publish your finished application on the Google Play Store, you must generate a signed APK (the Android application package).
- The APK is the compiled, executable version of your application.
- Signing it is much like signing your name to a document. The signature identifies the app's developer to Google and the users who install your application.
- More importantly, unless your Android Studio is in developer mode, unsigned applications will not run. Use the following steps to generate a signed APK:

1. Generate a signed APK from your code by selecting Build ➪ Generate Signed APK from the Menu bar to bring up the Generate Signed APK window



2. Assuming you have never published an application from Android Studio, you need to create a new key store. Click the Create New button to display the New Key Store window.

3. Fill out all of the information on this form because it pertains to your entity and application. Notice that there are two places for a password. These are the passwords for your key store and your key, respectively. Because a key store can hold multiple keys, it requires a separate password than that of the key for a specific app.



4. Click OK to return to the Generate Signed APK window.
5. In the Generate Signed APK windows, click Next to review and finish the process.Now that you have a signed APK; you can upload it to the Google Play Store using the developer console at**https://play.google.com/apps/publish/.**