Mobile Application Development

Model Paper - 3

Section - A

1) what is the Android Manifest file used for?

ans: The manifest file is an important part of our app because, it defines the structure and metadata of our application, it components, and its requirements. This file includes nodes for each of an Activities, Services, Content providers and Broadcast receiver that make the application and using Intent filters and Permissions, determines how they co-ordinate with each other and other application.

2) List two methods for debugging an Android application.

ans: ① Temporary Breakpoints

A temporary breakpoints is useful when you are trying to debug a large loop, or you just want to make sure a line of code is being hit during execution.

② Conditional Breakpoint

It is a breakpoint at which Android studio only pauses when specific conditions are met.

3) Name two basic views in Android.

ans: ① TextView View
Used to display text to the user

② Button View
Represents a push button widget

4) what is a fragment in Android?

ans: A fragment represents a modular portion of the user interface within an activity. A fragment has its own lifecycle, receives its own input events, and you can add or remove it fragments while the containing activity is running

5) Mention two ways to manage screen orientation changes.

ans: ① Creating an application that displays message based on the screen orientation

Create project : App → manifest
→ AndroidManifest.xml

code : android : configChanges = "orientation|screenSize"

② Inside MainActivity.java
=> Enter ctrl+O, select, onConfigurationChanged.
Then add following code:

```
if (newConfig.orientation == Configuration.ORIENTATION_
                    LANDSCAPE) {

Toast
toast = Toast.makeText(this, "orientation_landscape",
                    Toast.LENGTH_SHORT);
```

```
        Toast.show()
    }
    else if(newConfig.orientation == Configuration.
                ORIENTATION_PORTRAIT){
Toast toast = Toast.makeText(this, "orientation -
    Portrait", Toast.LENGTH_SHORT);
    toast.show();
    }
}
```

6) what file format is used for Android app deployment?

ans: An APK file (Android Package Kit file format) is the file format for applications used on the Android Operating System (OS)

Section B

7) Describe the steps involved in creating an Android project in the IDE.

ans: Creating an Android project in an Integrated Development Environment (IDE) such as Android Studio involves several steps:

① Install Android Studio using correct instructions

② Launch Android Studio
Open Android studio after installation is complete

③ Start a New Project
On welcome screen, click on "Start a new Android Studio project".

④ Configure Your Project
→ Name the application, package format, com.example.myapp, save location · choose the directory to save the project, select the language, choose the minimum Android version.

⑤ Select a project template
Choose a new to template like Empty Activity, Basic Activity, Navigation Drawer Activity etc and click on 'Next'

⑥ Configure Activity
Default android name will be 'Activity_Name', layout name 'activity_main', generate layout file option should be checked. for compatibility use a legacy support library

⑦ Click on 'Finish' & project will be created

⑧ Wait for project setup
Android studio will now build the project

⑨ Explore Project structure
located in the project window on the left which includes manifest, java, and res etc. Gradal scripts are used to managing dependencies & build configurations "build.gradle" file is located in the app directories

⑩ Run Your App
Now write the code, design UI and add dependencies. Connect physical device or AVD to run the program

⑪ Test the App
Use built in Android Emulator or physical device and debugging devices

⑫ Build & Release
After completion, use Build > Generate Signed Bundle/APK menu to prepare your app for release

8) Explain the use of Intents for inter-component communication in Android.

ans: An Android Intent is an abstract description of an operation/action to be performed

Intents are used to:
-> Start the service
-> Launch an activity
-> Display a web page
-> Display a list of contacts
-> Broadcast a message
-> Dial a phone call etc

Types of Intents

1) Explict Intent

Explict Intents are communicated between two activities inside the same applications. We can use explicit intents when we need to move from one activity to another activity

Eg:- when a user wants to start a service to download a file or when a new activity gets started in response to a user action

```
=> intent i = new Intent (getApplicationContext(),
                          ActivityTwo.class);
```

startActivity(i):
-> In Explicit we use the name of the component which will be affected by Intent
-> For Example, If we know the class name then we can navigate the app from one activity to another activity using Intent. In the similar way, we can start a service to download a file in a background process.

→ Explicit Intent works internally within an application to perform navigation and data transfer.
→ Explicit Intent specifies the component. In such case, intent provides the external class to be invoked.

② Implicit Intent

In Implicit Intents we do need to specify the name of the component. We just specify the action that has to be performed and further, this action is handled by the component for another application.
→ The most basic example of implicit intent is to open any web page
→ intent i = new Intent(android.content.Intent.ACTION_VIEW, Uri.parse("http://www.amazon.com"));
startActivity(i);
→ For some times, there is no need to specify the data. For example, to select a contact from the contacts application, you specify the action and then indicate the MIME type using the setType() method.

10) How can you programically create a user interface in an Android application?

ans: package com.example.programmaticui;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.LinearLayout;
import android.widget.TextView;
import androidx.appcompat.app.AppCompatActivity;
public class MainActivity extends AppCompatActivity {

```
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
    LinearLayout linearLayout = new LinearLayout(this);
linearLayout.setOrientation(LinearLayout.VERTICAL);
linearLayout.setLayoutParams(new LinearLayout.LayoutParams(
  LinearLayout.LayoutParams.MATCH_PARENT,
  LinearLayout.LayoutParams.MATCH_PARENT
    ));
    TextView textView = new TextView(this);
    textView textView =
    textView.setText("Hello, world!");
    textView.setLayoutParams(new LinearLayout.
                    LayoutParams(
LinearLayout.LayoutParams.WRAP_CONTENT,
LinearLayout.LayoutParams.WRAP_CONTENT
    ));
    Button.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v)
  {
      textView.setText("Button Clicked!");
            }
        });
    LinearLayout.addView(textView);
    LinearLayout.addView(button);
    setContentView(linearLayout);
  }
}
```

10) Discuss the role & usage of ActionBar in Android application.

ans: In Android application, ActionBar is the element present at the top of the activity screen

-> Besides fragments, another feature of Android is the Action Bar

-> In place of the traditional title bar located at the top of the device's screen, the Action Bar displays the application icon and the activity title

-> Optionally, on the right side of the Action Bar to the are action items

-> The setSupportActionBar() method writes your Action Bar to the screen. The Action Bar can be an instance of a Tool Bar.

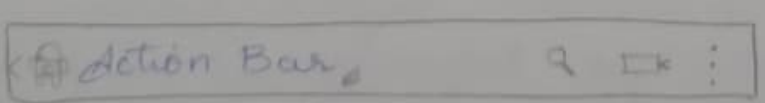
-> Components included in the Action Bar are:-

① App Icon - Display the branding logo/icon of the application

② View Controls: Section that displays the name of the application or current activity. Developers can also include spinner or tabbed navigation for switching between views

③ Action Button: Contains some important actions/elements of the app that may be required to the users frequently

④ Action Overflow: Include other actions that will be displayed as a menu. More less often used actions to the action overflow.



11) Outline the process for persisting user data using shared Preferences.

ans: ① Get the SharedPreference Instance
Obtain a reference to the SharedPreference object.

② Edit the SharedPreferences
Create an editor to modify the preferences

③ Add Data to sharedPreferences
Put data into the SharedPreferences using the editor

④ Save the Changes
Commit the changes using - commit() or apply()

⑤ Retrive Data from sharedPreferences
Fetch the stored data when needed

⑥ Remove Data from SharedPreferences
Remove specific data if no longer needed.

Key Points

-> File Name and Mode
SharedPreferences file name is a unique identifier. The mode can be 'Context.MODE_PRIVATE', ensuring that the preferences are accessible only by your application.

-> Editor
Use the 'SharedPreferences.Editor' to make changes. apply() saves the changes in the background, while commit() writes them immediately.

-> Default Values
When retrieving data, provide default values in case the key does not exist.

12) What are the steps to implement an about box in an Android application?

ans: Step 1: Design the Layout for the About Box.

Create the xml layout, design the UI layout for the "About" dialog using xml file. This layout might include TextViews for the app version, developer information and any other relevant details, as well as optional buttons or links.

Step 2: Create a DialogFragment class:

Create a DialogFragment class, define a new class that extends DialogFragment. This class will be responsible for displaying the 'About' dialog using the layout designed in step 1.

Override onCreateDialog method, In this class, this method is overridden to build & return an AlertDialog using the layout resource. Set the dialog title, content view, and any buttons like 'OK'.

Step 3: Add a Menu Item or Button to Trigger the About Box

Create a Menu Resource file by defining a new xml file in the res/menu directory to specify the menu items. This file will include an item for the "About" dialog.

Inflate the Menu in MainActivity by Overriding the onCreateOptionsMenu method in your main activity to inflate the menu resource. This makes the menu item visible in the activity's action bar or overflow menu

Step 4 : Handle Menu Item or Button Click Event

Override onOptionsItemSelected - In your main activity, override the onOptionsItemSelected method to handle the selection of the "About" menu item. This method will create an instance of the DialogFragment and display it.

Section C

13) Explain the process of creating and using an SQLite database in an Android Application-

ans: Creating & using an SQLite database in an Android application involves several well-defined steps, which include setting up the environment, defining the database schema, implementing CRUD operations, and managing the database lifecycle.

① Setup : Include Necessary Dependencies

SQLite is built into Android, so you don't need to add any additional dependencies in build.gradle file. It's part of the Android SDK.

② Database Creation : Create Database Helper class.

Subclass SQLiteOpenHelper to manage database creation & version management.

This class requires you to override two key methods.

OnCreate and onUpgrade

③ Using the Database : Initialize the Database Helper

Create an instance of your MyDatabaseHelper class to get a reference to the database. This is typically done in an Activity or ViewModel

④ Data Manipulation : Insert Data

Use ContentValues to insert data into the database. This method is safe & prevents SQL injection.

- Query Data

Retrieve data using the query method. This method provides a flexible way to retrieve data from the database

- Update Data

Use the update method to modify existing records in the database

- Delete Data

Remove records using the delete method

⑤ Management

- Handling Database Upgrades

When you need to update the database schema, increment the DATABASE_VERSION and modify the onUpgrade method accordingly

- Closing the Database

Always close the database and cursor objects to free up resources and avoid memory leaks.

⑥ Best Practices

- Use transaction for batch operations to improve performance

- Close the cursor after querying data to release resources
- Perform database operations on background threads to prevent blocking the UI thread
- Implement proper error handling to manage potential SQLiteExpr SQLiteException scenarios

14) ~~Describe~~ Describe how to use specialized fragments and their benefits in Android development.

ans: Fragments are mini-Activity that have their own life cycle.
There are 3 sub-classes of 'fragments'.

① List Fragment
→ This fragment contains a list view, which displays a list of items from a data source, such as an array or cursor
→ It is useful because, it is common to have one fragment that contains a list of items, and another fragment that displays details about the selected posting
→ List fragment could be used to display a list of messages exchanged between users in a chat conversation. Each item in the list represents a message, showing the sender's name, message content, timestamp, and possibly other metadata such as profile picture
→ To create a list fragment, we need to extend the ListFragment base class.

② Dialog Fragment

→ Dialog fragment floats on top of an activity and is displayed modally

→ It is useful when you need to obtain the user's response before continuing with execution

→ To create a dialog fragment, you must extend the DialogFragment base class

③ Preference Fragment

→ In android application, preferences are provided to the user to personalize the application

→ In Android, we can use 'PreferenceActivity' base class to display an activity for the user to edit the preferences. In Android 3.0 and later, we can use the PreferenceFragment class to do the same thing.

→ There are different types of preferences:

- EditTextPreference:
  This is used to display get the text from the user
- ListPreference:
  This option is used to display a dialog with the list of options to choose from.
- CheckBoxPreference:
  This option is used to display a checkbox to toggle a setting
- SwitchPreference:
  This option is used to turn the switch on and off

- Ringtone Preference :
  This option is used to open the ringtone page of your device
- Preference with an Intent action . VIEW - to open an external browser navigating to an URL

15) Discuss the implementation of SMS messaging in Android including sending & receiving messages.

ans : ① Sending SMS Messages

To send SMS messages, the SmsManager class is used. Here are some steps involved.

- Request Permissions :
  You need to request the necessary permissions in AndroidManifest.xml file.

```
<uses-permission
 android:name = "android.permission.SEND_SMS"/>
<uses-permission
 android:name = "android.permission.RECEIVE_SMS"/>
<uses-permission
android:name = "android.permission.READ_SMS"/>
```

- Sending SMS
  Use the SmsManager class to send SMS messages
  -> phoneNumber : The recipient's phone number
  -> message : The text message to be sent.
  -> The null values are for scAddress, sentIntent, and deliveryIntent, which can be used for more advanced SMS sending with status reports
- Runtime Permissions Handling
  For runtime permissions, you can check and request permissions as follows.

② Receiving SMS Messages

To receive SMS messages, we need to implement a BroadcastReceiver that listens for incoming SMS messages.

- Define the BroadcastReceiver
  Create a class that extends BroadcastReceiver
- Register the BroadcastReceiver
  Register the BroadcastReceiver in AndroidManifest.xml file

③ Permissions & Security Considerations

- Handling Permissions
  Ensure that you handle permissions correctly, especially for device running Android 6.0 and above. Always check for permissions before performing SMS operations.
- Security Best Practices
  → Be cautious with the data you send via SMS, as it is not encrypted
  → Validate and sanitize any user input that may be sent via SMS to avoid injection attacks

14) Explain how to consume web services using HTTP and JSON in android applications.

ans: ① Setup

- Add Internet Permission
  Ensure that your application has the permission to access the internet. Add the following permission to your AndroidManifest.xml file

```
<uses-permission
android:name="android.permission.INTERNET"/>
```

④ Making HTTP Requests

• using HttpURLConnection :

HttpURLConnection is a class provided by Android to handle HTTP connections.

→ urlString : The URL of the web service
→ method : The HTTP method (GET, POST, etc)
→ payload : The JSON payload for POST requests

• using OkHttp Library

OkHttp is a third party library that simplifies HTTP communication in Android. Add the dependency in your build.gradle file.

∴ implementation
'com.squareup.okhttp3:okhttp:4.9.1'

Then, use OkHttp to make HTTP requests.

⑤ Parsing JSON Response

• using JSONObject and JSONArray

Once you have the JSON response as a string, you can parse it using Android's JSONObject and JSONArray classes.

⑥ Async Task for Network Operations

Network operations must be performed on a background thread to avoid blocking the UI thread. Use AsyncTask or similar concurrency frameworks like ExecutorService or Coroutine for this purpose

→ To execute the AsyncTask :

new
NetworkTask().execute("https://api.example.com/data", "GET");

14) Describe the steps involved in creating & publishing a content provided in Android.

ans: Creating & publishing a Content Provider in Android involves several steps to properly define, implement and expose the provider to other applications or components within the same application.

① Define the Content Provider
→ Create a class Extending, ContentProvider
- Create a new Java or Kotlin class that extends ContentProvider.
- Implement the required methods:
- onCreate(): Initialize your database or other data storage mechanism
- query(): Handle query requests from clients
- insert(): Handle requests to insert new data
- update(): Handle requests to update existing data
- delete(): Handle requests to delete data
- getType(): Return the MIME type of data for the given URL

② Implement Database Helper
If we are using an SQLite database create a helper class to manage database creation & version management, typically SQLiteOpenHelper

③ Register the Content Provider in Manifest

Declare your Content Provider in the AndroidManifest.xml file, specifying its authority (a unique identifier) and whether it should be accessible from other applications

```
<provider
    android:name=".MyContentProvider"
    android:authorities="com.example.mycontentprovider"
    android:exported="true" />
```

- android:authorities: A unique string that identifies the content provider
- android:exported: Indicates whether the provider is available to other applications

④ Define URI Matcher and MIME Types

To distinguish between different types of URIs, use UriMatcher. Define constants for different URI patterns and register them in a static block

⑤ Implement CRUD Operations

Implement the query, insert, update and delete methods using your database or other data storage mechanism

- Query method

Handle query requests and return a cursor to the result set.

- Insert method

Handle insert requests and return the URI of the newly inserted row.

- Update Method
  Handle update requests & return the number of rows affected
- Delete Method
  Handle delete requests & return the number of rows deleted

⑤ Testing the Content Provider
- Use Android's ContentResolver to interact with the Content Provider
- Perform CRUD operations using methods like insert(), query(), update(), and delete() on the ContentResolver

18) Discuss how to handle local data access, including internal internal file system and SD card access in an Android application.

ans: Handling local data access in an Android application involves interacting with both the internal file system & the SD card (extra storage). Each storage type has specific use cases & access methods, which are crucial for data management within the application.

① Internal Storage
  Internal storage is private to the application, meaning that files stored here are not accessible by other applications & users.
  This is suitable for storing sensitive data or data that should remain private to the app.

- File handling in Internal Storage
  - => Writing to Internal Storage
    - -> use the openFileOutput() method to write a file
    - -> this file will be created in the app's private storage directory
  - => Reading from Internal Storage
    - -> use the openFileInput() method to read from a file
  - => Deleting a file
    - -> use the deleteFile() method to delete a file from internal storage
- Storing in SharedPreferences

```
String filename = "myfile";
boolean deleted = deleteFile(filename);
```

- Storing in SharedPreferences

SharedPreferences is used for storing small amount of data in key-value pairs

  - -> Writing to SharedPreferences
  - -> Reading from SharedPreferences

② External Storage (SD card)

External storage is accessible by the user and other applications, making it suitable for large files and data that should be shared or backed up. This includes the SD card or other shared storage locations.

-> Permissions

Starting with Android 6.0 (API level 23), you need to request runtime permissions for accessing external storage

- Manifest Permissions
- Requesting Permissions at Runtime

-> File Handling in External Storage

- Check storage Availability
- Writing to External Storage
- Reading from External Storage
- Deleting a file

(3) Using MediaStore for media files

The MediaStore API provides a structured interface for storing and retrieving media files, such as images, audio & video, which are accessible to all applications.

-> Saving an Image to MediaStore

-> Querying MediaStore for Media files