Date __/__/____
Page _____

Mobile Application Development
Model Paper - 02

* 2 Marks (PART - A)

① Define Dalvik Virtual Machine?

Ans The Dalvik Virtual Machine (DVM) is an integral part of an Android Operating System. It is a specialized virtual machine designed by Google for Android to run applications written in Java.

② What is Menu?

Ans In Android Menu is a user interface that provides actions & options that users can choose from. It typically appears as a list of items displayed on the screen, allowing users to navigate through different parts of application or perform specific tasks.

③ List the types of Mobile technology?

Ans
1. Native Development
2. Cross Platform Development
3. Hybrid Development
4. Progressive Web APPs

④ What are the two attributes of Image View?

Ans In MAD, There are several attributes of ImageView that can be used to control the appearance & behavior of an image within the app.

⇒ Some common attributes are:

① android:src
② android:background
③ android:onClick
④ android:ScaleType.

---

⑤ What is GPS?

Ans: GPS (Global Positioning System) is a satellite-based system that provides location & time information anywhere on Earth.

Ex:- • Google Maps
• Apple Maps.

---

⑥ List the steps to create database Cursors?

Ans
1. Open or create a Database
2. Query the Database
3. Retrieve Data from the Cursor
4. Iterate through the Cursor
5. Close the Cursor.

Date __/__/____
Page _____

5 Marks (PART - B)

(7) What are the types of Basic Views?

Ans:- In Android, Views are fundamental building blocks used to create the UI of an application.

Some commonly used Basic Views are.

1. Text View
2. Edit View
3. Button
4. Image Button
5. CheckBox
6. Toggle Button
7. RadioButton
8. ProgressBar
9. AutoComplete Text View.

① **TextView:**

It is the one of the most fundamental & commonly used Views in Android app. TextView in Android is used to display text to user.

② **EditView.**

Edit View is a key component in Android development, designed to allow users to input & modify the text.

③ **Button:**

Button is a fundamental UI Component in Android development & it is designed to trigger actions when clicked.

④ **Image Button.**

Image Button is a fundamental UI Component in Android development, designed to Provide a clickable button with an image.

⑤ CheckBox:

CheckBox is a UI component in Android that Allows users to make a binary choice, typically indicating a choice between True & false.

⑥ Toggle Button:

Toggle Button is a UI Component in Android that allows users to switch between two States: Checked & Unchecked.

⑦ RadioButton:

RadioButton is a UI Component in Android that allows users to select one Option from a set of mutally exclusive options.

⑧ ProgressBar:

Progress Bar is a UI Component in Android that is commonly used to Show the status of ongoing operations such as loading data, downloading files or processing tasks.

⑨ AutoCompleteTextView:

AutoCompleteTextView is a UI Component in Android that provides Suggestions to users as they type.

---

⑧ Explain Content Provider?

Refer Model Paper - 01 Ques No - 12.

Date __/__/____
Page _____

9. Explain fragments & intents?

Ans
- fragments in Android are modular & reusable UI components that represent a portion of a user interface or behavior within an activity.
- They are used to build dynamic & multi-pane layouts that can adapt to various device configurations.
- It is especially useful for applications that runs on variety of screen sizes such as smartphones & tablets.

⇒ features & Benefits of fragments.

① Modularity:
fragments allow the UI to be broken down into smaller, manageable piece

② Reusability:
fragments promote code reuse. Once a fragment is created, it can be used across multiple activities & different parts of application.

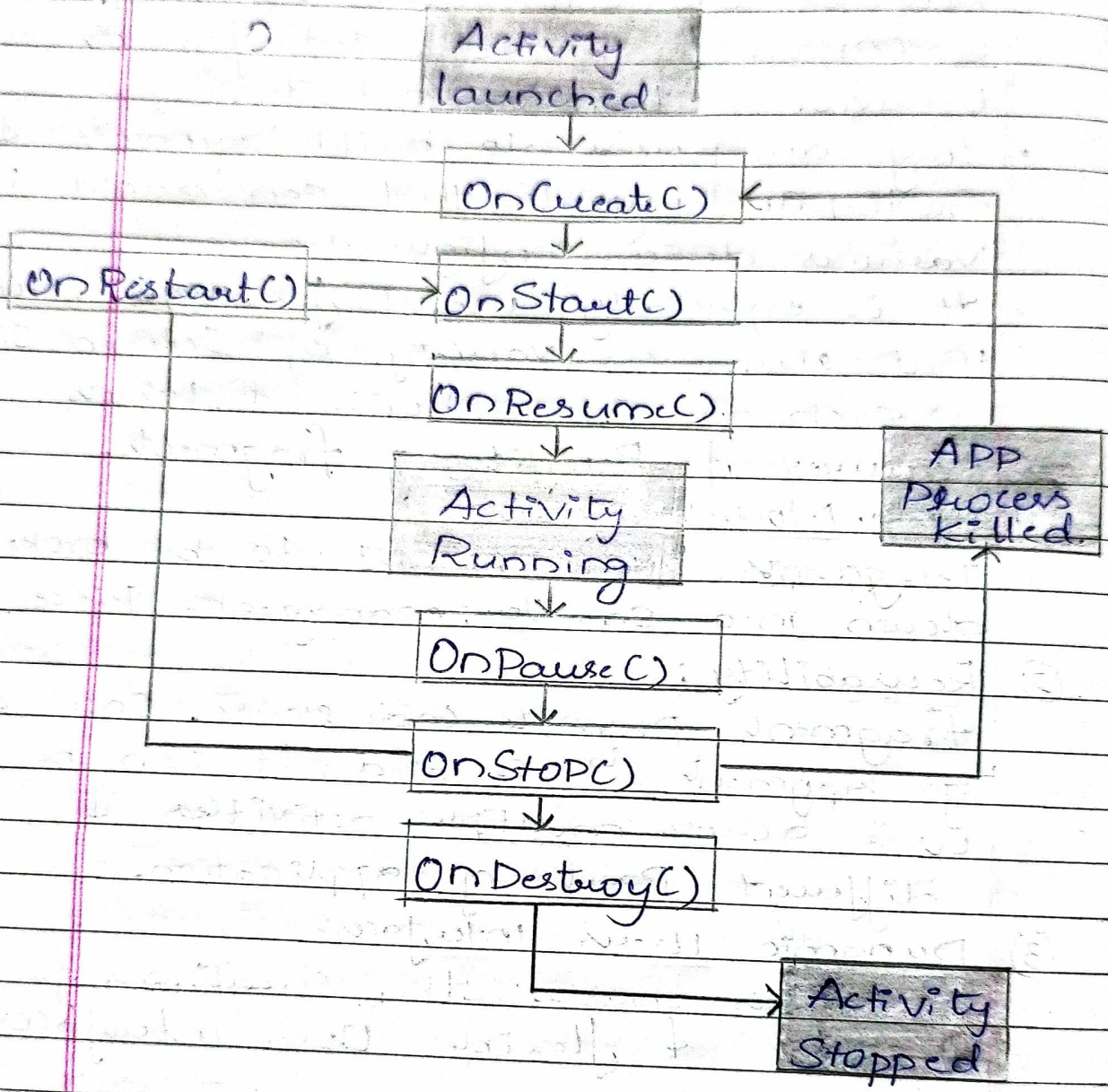
③ Dynamic Users Interfaces:
fragments enables the creation of dynamic & flexible User Interfaces.

④ Communication:
fragments can communicate with their host activity & other fragments.

⇒ Intents
Refer Model Paper - 01 Q.No ⇒ 13.

10. Explain the life cycle of Android activity?

Ans



① **OnCreate():**
It is called when the activity is just created.

② **OnStart():**
It is invoked when the activity is visible to the users.

Date __/__/____
Page _____

③ OnResume():
It is invoked when the activity starts interacting with users.

④ OnPause():
It is invoked when the system is about to start resuming another activity.

⑤ OnStop():
It is invoked when the activity is no longer visible to users.

⑥ OnDestroy():
It is invoked ~~when~~ before the activity is destroyed

⑦ ~~On~~ OnRestart(): OnRestart():
It is invoked when the activity has been stopped & is restarting again.

---

⑪ Explain to apply a Style and theme?

Ans The Styles & themes are Powerful tools to define & apply consistent visual Styles across an application or Specific components like activities.

⇒ Styles:

Definition: A Style in Android is a collection of properties that Specify the look & format for a View or a window

Usage: Styles can be applied to individual views in layout XML files in Java / Kotlin code.

Purpose: • Reusability
• Customization.

Components: • Style Resource
• Attributes.

=> Themes:

Definition: A Theme in Android is a style applied to an entire Activity or application rather than an individual views.

Purpose: • Global Consistency.
• Customization.

Components:
• Theme Resource.
• Inheritance.

Application level vs. Activity level:
Themes can be applied at the application level in androidManifest.xml file or at the activity level in layout XML files.

Customization:
Themes can be customized to create unique visual experience for different parts of application.

Date __/__/____
Page _____

(12) Explain menus & additional Views?

Ans ⇒ menus.

Refer MP - 02 Question NO - 02 & MP - 01 Q - NO - 7.

⇒ Additional Views:

Refer MP - 02 Q.NO - 07 (Basic Views)

---

*(13) 8 Marks (PART - C)

---

(13) Explain Date Picker, Time Picker & List View?

Ans ⇒ Date Picker:

- It is a UI Component that allows users to select a specific date.
- It can display dates in various formats & allows navigation between days, months & years.

* features of Date Picker.

1. Date Selection: Allows users to select a specific date from a calendar or Spinner View.
2. Customizable Display: Can be displayed as a calendar or spinner, depending on the mode set.
3. Min & Max Date: Supports setting minimum & maximum selectable dates to restrict user input to a specific range.
4. OnDateChangedListener: Allows developers to handle date changes & perform actions when the selected date changes.

⇒ Time Picker:

- It is a UI component that allows users to select a specific time.
- It can be displayed in either 12-hour or 24-hour formats.

* Features of Time Picker:

① Time Selection: Allow users to select a specific time of day.

② 12-Hour & 24-hour format: Can be configured to display time in either 12-hour (AM/PM) or 24-hour format using is24HourView attribute.

③ OnTimeChangedListener: Allow developers to handle time changes & perform actions when the selected time changes.

⇒ List View:

It is a fundamental component in Android development that allow developers to display a scrollable list of items on the screen.

* Features of ListView:

① Efficient Data Management:
'ListView' handles large dataset efficiently by reusing view elements that have gone off-screen, a process known as View recycling.

② Customizable Item Layouts: Each item in a 'ListView' can be customized using XML layout files. Developers can define complex item layouts that includes various UI elements.

Date __/__/____
Page _____

③. Adapters: The 'ListView' relies on adapters to bind data to the View. It act as intermediaries b/w the data source & 'ListView', converting data items into Viewable components.

④. Event Handling: 'ListView' provides built-in support for handling user interactions such as item clicks, long presses & contextual menus.

14. What are Android application components?

Ans Android components can be categorized into primary components (core components) & Secondary components (non-core components) based on their fundamental roles & supplementary functionalities within an application.

1. Primary or Core components:
Core Components are crucial for the basic functionality & structure of an Android Application.
Key features.

a Activities: It represent the UI of an application. Each screen in an Android app is implemented as an activity.

b. Services: It is a component that performs long ~~term~~ running operation in background without UI.

c Content provider: It manage a shared set of app data. They allow apps to securely share data with other apps or access data from other apps.

2. Secondary or Non-core Component :-
Non-core components enhance the primary functionalities & improve user experience, but they are not essential for the basic operation of app.
Key features:
a. Fragments: It represents a behavior or a portion of the UI in activity. They can be dynamically added, removed or replaced within an activity.
b. Intents: Intents are messaging objects used to request actions from other app components. They can carry data to be used by the target component.
c. Adapters: It acts as the bridge b/w UI components & datasources to facilitate the display of data in UI components.

15. Explain the creation of login window?
Ans Refer 3rd program from Record.

16. Explain Android Studio Components?
Ans Refer MP-02 Q-NO-16
- Activities
- Services
- Content Provider
- fragments
- Intents
- Adapters

Date __/__/____
Page _____

17. What is the use of APK file?

Ans. APK file plays a crucial role in mobile Application Development. APK stands for Android Package kit.

⇒ Uses of APK files are:

1. Distribution:
   APK files are used to distribution Android apps to users through platforms like the Google play store.
2. Installation:
   Users download & install APK files to add new apps or update existing ones on their Android devices.
3. Testing:
   Developers use APK files for testing apps on different devices to ensure compatibility & functionality.
4. Security:
   APK files are signed with digital certificates to verify the authenticity of the app & prevent tampering.
5. Updates:
   Developers release new versions of APK files with bug fixes, new features, or improvements for users to download & install
6. Offline Installation:
   APK files allow users to share apps offline with others without requiring an internet connection.

Create an application to send SMS and receive SMS

Steps:

1. Click **Start-** Android **Studio**, a **Welcome to Android Studio** dialog box will appear. Click **New Project**, the **New Project Dialog box** appears.
2. Choose **Empty Views Activity** then click **Next**.
3. Specify the **Name** of your project, Select the **Language** as **Java**, and Select the **SDK as API 24("Nougat",Android 7.0).**Click **Finish** Button.
4. Update the following code **in activity_main.xml and MainActivity.java**
5. Click **Run app** or **shift+F10** to execute the application.

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/white" tools:context=".MainActivity">
    <EditText
        android:id="@+id/editTextPhoneNumber"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Enter phone number"
        android:layout_margin="16dp"/>
    <EditText
        android:id="@+id/editTextMessage"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Enter message"
        android:layout_below="@id/editTextPhoneNumber"
        android:layout_margin="16dp"/>
    <Button
        android:id="@+id/buttonSend"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Send"
        android:layout_below="@id/editTextMessage"
        android:layout_alignParentEnd="true"
        android:layout_marginEnd="16dp"
        android:onClick="sendMessage" tools:ignore="UsingOnClickInXml" />
```

```
    <TextView
        android:id="@+id/textViewReceivedMessages"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/buttonSend"
        android:layout_marginStart="16dp"
        android:layout_marginTop="16dp"
        android:layout_marginEnd="16dp"
        android:layout_marginBottom="16dp"
        android:textColor="@color/black" />
</RelativeLayout>
```

MainActivity.java

```
package com.bca.sms;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.content.pm.PackageManager;
import android.os.Bundle;
import android.telephony.SmsManager;
import android.telephony.SmsMessage;
import android.view.View;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;
import android.Manifest;
import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity
{
```

```
    private static final int SMS_PERMISSION_CODE = 101;
    private EditText editTextPhoneNumber;
    private EditText editTextMessage;
    private TextView textViewReceivedMessages;
    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        editTextPhoneNumber = findViewById(R.id.editTextPhoneNumber);
        editTextMessage = findViewById(R.id.editTextMessage);
        textViewReceivedMessages = findViewById(R.id.textViewReceivedMessages);
// Request SMS permissions if not granted

        if (!checkSMSPermission())
        {
            requestSMSPermission();
        }
// Register SMS receiver
```

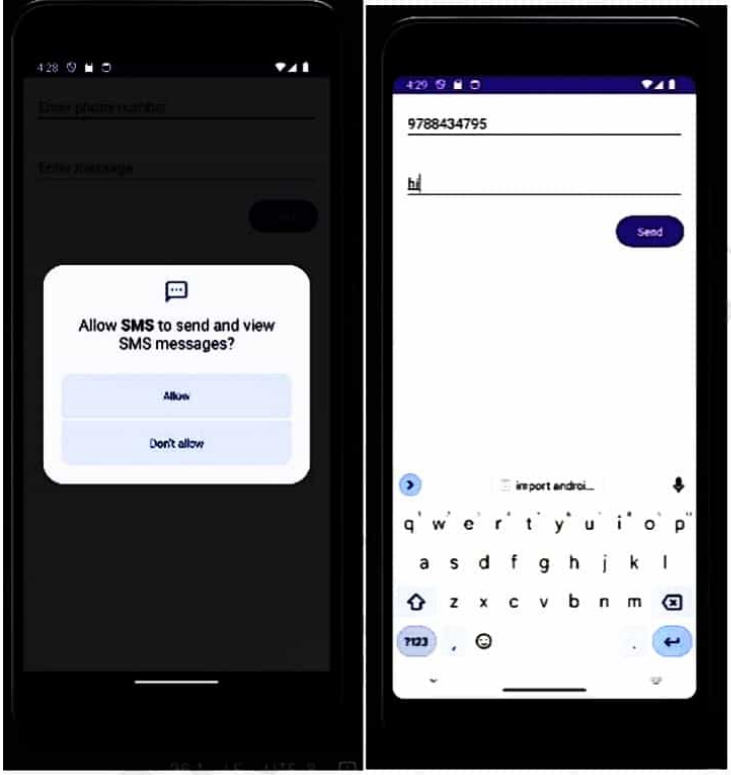
```
// Register SMS receiver
IntentFilter intentFilter = new IntentFilter();
        intentFilter.addAction("android.provider.Telephony.SMS_RECEIVED");
        registerReceiver(smsReceiver, intentFilter);
    }
    @Override
    protected void onDestroy()
    {
        super.onDestroy();
        unregisterReceiver(smsReceiver);
    }
    // Button click listener for sending SMS
    public void sendMessage(View view) { String phoneNumber =
            editTextPhoneNumber.getText().toString().trim();
        String message = editTextMessage.getText().toString();
        if (phoneNumber.isEmpty())
```

```
        {
            Toast.makeText(this, "Please enter a valid phone number",
Toast.LENGTH_SHORT).show();
            return;
        }
        try {
            SmsManager smsManager = SmsManager.getDefault();
            smsManager.sendTextMessage(phoneNumber, null, message, null, null);
            Toast.makeText(this, "Message sent", Toast.LENGTH_SHORT).show();
        }
        catch (IllegalArgumentException e)
        {
            Toast.makeText(this, "Invalid phone number format",
Toast.LENGTH_SHORT).show();
        } catch (Exception e) {
            Toast.makeText(this, "Failed to send message", Toast.LENGTH_SHORT).show();
            e.printStackTrace();
        }
    }

// Check if SMS permission is granted
    private boolean checkSMSPermission() {
        return ContextCompat.checkSelfPermission(this, Manifest.permission.SEND_SMS) ==
PackageManager.PERMISSION_GRANTED;
    }
    // Request SMS permission
    private void requestSMSPermission() { ActivityCompat.requestPermissions(this, new
            String[]{Manifest.permission.SEND_SMS}, SMS_PERMISSION_CODE);
    }
// SMS receiver
    private final BroadcastReceiver smsReceiver = new BroadcastReceiver()
    { @Override
    public void onReceive(Context context, Intent intent) { Bundle bundle = intent.getExtras();
        if (bundle != null)
```

```
{
        Object[] pdus = (Object[]) bundle.get("pdus");
        if (pdus != null) {
           for (Object pdu : pdus)
 {
                SmsMessage smsMessage = SmsMessage.createFromPdu((byte[]) pdu);
                String senderPhoneNumber = smsMessage.getDisplayOriginatingAddress();
String messageBody = smsMessage.getMessageBody();
                textViewReceivedMessages.append("From: " + senderPhoneNumber + "\n");
                textViewReceivedMessages.append("Message: " +
                     messageBody + "\n\n");
           }
        }
     }
   }
   };
}
```

Output


Allow SMS to send and view SMS messages?
Allow
Don't allow
9788434795
hi
Send


9788434795
hi
Send