MOBILE APPLICATION DEVELOPMENT

MODEL PAPER 3

SECTION - A

1. What is an APK?

APK (Android Package): APK stands for Android Package Kit. It is the file format used by the Android operating system for distribution and installation of mobile apps and middleware. An APK file contains all the elements that an app needs to install correctly on a device, including the app's code, resources, assets, certificates, and manifest file. When developers build an Android app, the Android SDK tools compile the code and package it into an APK, which can then be tested, shared, or uploaded to app distribution platforms like Google Play. For example, when building a simple calculator app, the resulting APK contains the app's code, images, and layout files.

2. What is an Activity in Android?

An activity in Android is a single screen with the user interface where users can interact it represents a window in an application that displays the UI elements it's screen in an android app is typically implemented as an activity

3. What is a testView in Android?

The text we in Android is used to display text to the user it is one of the most fundamental and commonly used wheels in Android app development the text view component that can be style and customised to fit various design requirements for example we can italicise or bold specific parts of the text to highlight important information additionally if the text contains a hyperlink directing to a specific web URL we can span it with the hyperlink and underline it.

4. What is a ListView in Android?

Listway visa fundamental component in Android development that allows developers to display is cruelble list of items on the screen it offers various features and functionalities to enhance the

User experience and facilitate their presentation of data in a structured man it is ideal for presenting large data sets in a structure format to navigate an interact with the data efficiently and intuitively. The list where is particularly beneficial when the number of items to be displayed is not known before hand or when the data is dynamic.

5. What is Data Binding in Android?

Data Binding in Android refers to the technique of establishing a connection between the UI elements such as text view edit text recycular wheels in the layout files and the data sources like variables objects in the application code this binding is set up using binding expressions in XML layout files which define how the UI element should display or interact with the data.

6. What do you mean by XML Web Services

XML extensible markup language web services are a type of web service that uses XML as the format for transmitting data between a client and a server XML is a flexible structure data format that is both human readable and machine readable it is widely used in web applications for data exchange due to its flat form and language independence

Section-B

7. Explain Mobile Operating Systems.

Mobile Operating Systems (05) are the software platforms that manage the operations of mobile devices such as smartphones, tablets, and smartwatches. Mobile operating systems serve as the backbone of smartphones by providing the essential software framework that enables devices to function, run applications, and connect to services. These 05 provide the basic functions necessary for the device to operate and for users to interact with the device and its applications

Some Key Mobile Operating Systems:

+ Android: Developed by Google, Android is the most widely used mobile operating system

globally. It is based on the Linux kernel and is known for its open-source nature, allowing

manufacturers to customize it for their devices. Android has a vast ecosystem of apps available

through the Google Play Store

+ 10S: Developed by Apple, IOS is the operating system used exclusively on Apple's mobile devices including the iPhone, iPad, and iPod Touch. iOS is known for its smooth performance, security features, and tight integration with Apple's hardware and software ecosystem. Apps for iOS devices are available through the Apple App Store

+ Others: While Android and iOS dominate the mobile operating system globally, there are other operating systems that have been used in mobile devices These include now-discontinued platforms like Windows Mobile and BlackBerry OS, as well as niche platforms like Tizen and HarmonyOS.

Windows Mobile: Once a popular choice for business users due to its integration with Microsoft Office and Exchange Server, Windows Mobile has been discontinued. Microsoft shifted its focus to developing applications for Android and iOS.

BlackBerry OS: Known for its strong security features and physical keyboards, BlackBerry OS was a favorite among enterprise or business users. However, it has been largely phased out, with BlackBerry devices now running on Android

Tizen: An open-source operating system developed by the Linux Foundation and supported by Samsung Tizen is used in a range of Samsung devices including smartwatches, TVs, and some smartphones, though it has not achieved widespread adoption.

HarmonyOS: Developed by Huawei, HarmonyOS is designed to work across a variety of devices, including smartphones, tablets, wearables, and lot devices. It represents Huawei's effort to reduce its dependency on Android amid geopolitical challenges.

B.Write the Steps to Debug an Android Application in Android Studio

1. Set Breakpoints:

One of the most powerful debugging features in Android Studio is the ability to set breakpoints in code. Breakpoints allow the app to pause at specific points to inspect the state of the app

To Set Breakpoints: Click in the left gutter next to a line of code to set a breakpoint. A red dot will appear

Manage Breakpoints: View and manage breakpoints via Run > View Breakpoints. Once a breakpoint is set, the app can be run in debug mode by clicking on the "Debug" button in Android Studio. When the app reaches the breakpoint, it will pause to inspect the values of variables, step through the code, and see the state of the app.

2 Connect a Device or Start an Emulator:

Before initiating the debugging process, developers need to connect a physical Android device to the computer via USB or start an Android Emulator from Android Studio's Device Manager This connected device or emulator will be used to run and debug the application.

3. Run the App in Debug Mode:

Developers can run the Android application in debug mode by clicking on the "Debug" icon (bug symbol) in the top toolbar of Android Studio or by using the shortcut Shift F9 They can select the connected device or emulator to deploy the application for debugging

4. Navigate Through the App:

Once the application is running in debug mode on the device or emulator, developers can interact with the app to trigger the execution of the code The application will pause at the set breakpoints to analyze the state of variables and understand the program flow

5. Inspect Variables and Evaluate Expressions:

Developers can utilize the Debug panel in Android Studio to inspect variable values, evaluate expressions, and monitor the application's state during runtime. This feature helps in understanding how data changes as the program executes

Hover over variables to see their values or view them in the Variables pane in the Debug tool window

Add variables to the Watches pane for continuous monitoring

Use the Evaluate Expression tool (accessible from the Debug tool window) to inspect and evaluate expressions during debugging

6. Step Through the Code:

Android Studio provides various debugging controls such as Step Over, Step Into, and Step Out to navigate through the code line by line. By stepping through the code developers can trace the execution path and identify any anomalies or errors

Use the stepping buttons in the Debug tool window to control code execution.

Step Over (FB): Moves to the next line of code

Step Into (F7): Enters the method on the current line.

Step Out (Shift+F8): Exits the current method and returns to the caller.

7. Evaluate Logs and Exceptions:

Developers should check the Logcat tab in Android Studio while debugging to view system logs, error messages, and exceptions thrown during the application's execution Analyzing logs

and exceptions helps in pinpointing issues and understanding the root cause of errors Open the Logcat tool window (View > Tool Windows > Logcat) to see log messages from the running app.

Use filters to narrow down logs by tag, level, or keywords.

8. Use Profiling Tools:

For advanced debugging and optimization, developers can leverage Android Profiler in Android

Studio. This tool provides insights into app performance, CPU usage, memory allocation, and

network activity, helping developers optimize their applications for better efficiency and user experience.
Use the Android Profiler tool (View > Tool Windows > Profiler) to analyze CPU, memory, and network performance.

Record and inspect method traces, memory allocations, and network activity

9. Modify Code and Re-Test:

Based on the insights gained from debugging, developers can make necessary code modifications to address issues. After making changes, developers should remove breakpoints, re-run the application in debug mode, and verify that the fixes have resolved the identified problems.

9. Explain Number Picker along with its attributes and features. Give an example

NumberPicker is a Ul component that allows users to select a number from a predefined range. It aplas a vertical list of numbers that can be scrolled.

Example Scenario: Selecting a quantity of items to purchase, setting a volume level, or choosing a

Features of Number Picker

1. Number Range Selection: Allows users to select a number from a predefined range using a vertical scrollable list.

2. Min and Max Values: Supports setting minimum and maximum values to define the range of selectable numbers

1. Customizable Formatter: Allows customization of the displayed values using a formatter.

4 Wrap Selector Wheel: Option to enable or disable the wrapping of the selector wheel. allowing values to wrap around from min to max and vice versa

10. Outline the steps to add an options menu in the Sudoku app, including changing the theme and defining the menu in XML

11. Explain the process of monitoring a location using the LocationManager class in Android

12. Explain how to retrieve data from a SQLite database using the query method

Section-C

13. Explain Different Mobile Technologies.

Different Mobile Technologies

1.3.1 Cellular Networks

1.3.2 Mobile Operating Systems (OS)

1.3.3 Mobile Development Frameworks

1.3.4 Mobile Web Technologies

1.3.5 Connectivity and Communication

1.3.6 Sensors and Hardware Innovations

1.3.7 Location-Based Services (LBS)

1.3.8 Cloud Storage

1.3.9 Emerging Technologies

1.3.10 Artificial Intelligence (Al) in Mobile

14.a) Explain the types of Intents with example code.

. Explicit Intents:

An explicit intent is used to launch a specific app component (eg a particular Activity or Service). We use explicit intents when we know the target component's class name.
This type of intent is particularly useful for invoking internal functionalities or features within the app that are well-defined and known at the development stage

Example Explicit Intents

Scenario: Viewing a Contact's Profile in WhatsApp

In WhatsApp, when a user taps on a contact's name to view their profile information, the apy uses an explicit intent to launch the "Profile Activity" within WhatsApp. This means that the app specifies the exact activity (ProfileActivity) that should be started to ensure that the aver taken directly to the contact's profile page within the app

Scenario: Initiating a Payment in Google Pay In Google Pay when a user selects a contact to send money to, the app uses an explicit intent start the "PaymentActivity". This intent includes details about the selected contact, such as thee name and payment information, and ensures that the "PaymentActivity" within Google Paya the one handling the transaction.

Scenario: Viewing Account Details in Banking App In a banking app, when a user clicks en one of their accounts to view more details, the app unes an explicit intent to start the "AccountDetailsActivity". This intent includes the specific account ID, ensuring that the "AccountDetailsActivity" displays the correct information for the selected account

Code

Explicit Intents

Intent intent new Intent (CurrentActivity this, ProfileActivity, class); Intent.putExtra("contact id", contactId);

startActivity(intent);

Explanation

This code snippet demonstrates the use of an explicit intent to navigate from the current activity to another specific activity (ProfileActivity) within the same app

Create Intent: An intent is created to transition from the current activity to ProfileActivity

Add Extra Data: Additional data (contact (D) is added to the intent, which can be used by

ProfileActivity to display specific information. Start Activity: The startActivity method is called with the intent, triggering the Android system

to start Profile Activity and passing the extra data along with it This approach ensures that when a user action (such as clicking on a contact's name) occurs, the app navigates to the correct activity and displays the relevant data for that specific contact

2. Implicit Intents:

Implicit Intents are used to request functionality from other components on the device witho specifying the exact component to be invoked. The Android system resolves the Intent to the appropriate component based on the Intent's action, data, and category

An implicit intent does not specify the component to start. Instead, it declares a general action

to perform, which allows a component from another app to handle it. The Android system

matches the intent to a suitable app component by comparing the intent to the intent files declared in the manifest file of other apps.
Scenario: Sharing a Location in WhatsApp

In WhatsApp, when a user wants to share their current location in a chat, the app uses an implicit intent to open the system's map application. This intent includes the location data but does not spectly which map app should handle it. The Android system then displays a list of all

installed map apps that can handle the location sharing, allowing the user to choose their preferred map App

Scenario: Scanning a QR Code

When a user chooses to scan a QR code for payment in Google Pay, the app uses an implicit intent to open any available QR code scanner app on the device. This intent specifies the action of scanning a QR code but does not specify a particular QR code scanner app The Android system presents a list of all apps that can perform QR code scanning, letting the user pick their preferred app.

Scenario: Making a Call to Customer Support

In a banking app, when a user clicks on a button to call customer support, the app uses an implicit intent to open the phone dialer This intent includes the customer support phone number but does not specify which phone app should handle the call The Android system presents the user with the phone dialer, allowing them to make the call using theit preferred phone app

Code

Impicit Intents

Intent intent new Intent(Intent.ACTION VIEW); intent.setData(Uri.parse("geo

: 37.7749, 122 4194")); if (intent.resolveActivity(getPackageManager()) != null) {

startActivity(intent);

Explanation

This code snippet demonstrates the use of an implicit intent to view a geographic location on a map

Here's the step-by-step process

1. Create Intent: An implicit intent is created with the action to view content.

2. Set Data: The data for the intent is set to a geographic location (latitude 37 7749, longitude -122-4194)

3. Check Availability: The code checks if there is an application installed on the device that can handle the intent to view the geographic location

4. Start Activity: If such an application exists, it starts the activity to view the location on a map. This approach allows users to view a specific location on a map by simply clicking on a link or button that triggers this intent. The Android system handles finding an appropriate map application that can display the location, making the process seamless for the user.

b) What is Dialog in Android? Explain the Types of Dialogs.

A dialog in Android is a small window that prompts the user to make a decision or enter additional interact with them before they can return to the parent activity. Dialogs are useful for

information Dialogs do not fill the screen and are usually modal, meaning they require the user to

Alerts: Informing users of events that need their attention.

Confirmations: Asking users to confirm an action

Inputs: Collecting user input without leaving the current context.

Types of Dialogs in Android

3.17

There are several types of dialogs in Android:

AlertDialog: AlertDialog is a versatile dialog that can show a title, message, and buttons. It is used for prompting user actions, displaying information, or gathering input.

DatePickerDialog: A dialog that allows users to select a date. It provides a user-friendly interface for date selection, showing a calendar view where users can pick a date.

TimePicker Dialog: A dialog that allows users to select a time It can be configured to use either a 24-hour format or an AM/PM format, depending on the application's requirements

Custom Dialogs: Fully customized dialogs using custom layouts Custom dialogs provide complete flexibility, allowing developers to design and implement any user interface within the dialog.

BottomSheetDialog: A BottomSheetDialog is a dialog that slides up from the bottom of the screen. BottomSheetDialog provides a modern way to present actions or options to users. often used for actions related to the current content.

15. Discuss how to create a preference management system using Shared Preferences. Provide a detailed code example

16.a) Explain Creating Separate Layouts for Different Orientations. How it works?

When a device changes or emotion, the streren dimensions and aspect ratio change. A layout designed for portrait mode may not look or function optimally in landscape mode and vice versa to addred this. Android allows to define different layout resources for different orientatione

?f What do you mean by Creating Seperate Layouts for Different Orientation 1

Creating separate layouts for different orientations is a technique used in Android development to provide customized user interfaces for portrait and landscape modes. By designing distinct layouts for each orientation, we can can optimize the Ul for the available screen space to ensure a better user experience

How It Works:

Android's resource system supports providingalternative resources for different device configurations, including screen orientation. By placing layout files in specific resource directories (res/layout for portrait and res/layout-land for landscape), the system automatically loads the appropriate layout based on the current orientation

1. Create Layout Files: Separate XML layout files for each orientation. For example, have activity main.xml for portrait mode and activity_main land xml for landscape mode

2. Organize Layout Resources: Place portrait layout files in the res/layout directory and landscape layout files in the res/layout-land directory. Android loads the appropriate layout based on device orientation.

3. Customize Layouts: Customize XML files to suit portrait and landscape orientations. Adjust element positions, sizes, and alignments as needed.

b) Explain Constraint Layout along with its features, attributes, advantages and disadvantage

Constraint layout is a powerful and flexible layout manager available in Android it allows to create large and complex layouts with a flat view hierarchy reducing the nesting view groups which can improve performance constant layout is part of the Android support library which makes it compatible with the wide range of Android versions.

Pavithra.S

A ConstraintLayout positions and sizes its child views using a set of constraints. These constraints rules that define the relationship of a child view to other views or to the parent layout. This allows Bexoble and responsive design as views can be positioned relative to each other or the parent conta in various ways.

Example Understanding Constraint Layout

In a room with furniture like chairs and tables, think of Constraint Layout in Android as a special tool th sets rules for the spacing between each piece of furniture and their alignment with walls or other items. For example, we can position a chair 20 inches away from a wall and align it with a table. If the table is used the chair will adjust its position automatically based on the rules set. This tool helps in creating a tidy an structured layout similar to arranging furniture in a room but on a digital screen.

Key Features

1. Flat View Hierarchy:

Reduces the need for nested view groups, leading to improved performance and simple layouts

2. Flexible Constraints:

Allows for complex positioning and alignment of views using constraints like 'start', 'n "top", "bottom", "baseline', and 'center'

3. Guidelines and Barriers:

Supports invisible guidelines to align views and barriers to create flexible group constrain

4. Dimension Constraints:

Supports 'wrap_content', 'match constraint' (Odp), and fixed dimensions, providing for control over view sizes.

5.Chaining:

Enable Changing of views horizontally or vertically to distribute space evenly and create Complex layouts with minimal effort.

Attribute and Description	Example
app:layout_constraintTop_toTopOf Positions the top edge of a view relative to another view or parent	app:layout_constraintTop_toTopOf="parent"
app:layout_constraintBottom_toBottomOf Positions the bottom edge of a view relative to another view or parent	app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintStart_toStartOf Positions the start edge of a view relative to another view or parent	app:layout_constraintStart_toStartOf="parent"
app:layout_constraintEnd_toEndOf Positions the end edge of a view relative to another view or parent	app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintHorizontal_bias Adjusts the horizontal bias (position) between start and end constraints	app:layout_constraintHorizontal_bias="0.5"
app:layout_constraintVertical_bias Adjusts the vertical bias (position) between top and bottom constraints	app:layout_constraintVertical_bias="0.5"
app:layout_constraintWidth_default Specifies the default width behavior, such as wrap or match constraint (0dp)	app:layout_constraintWidth_default="wrap"
app:layout_constraintHeight_default Specifies the default height behavior, such as wrap or match constraint (0dp)	app:layout_constraintHeight_default="wrap"
app:layout_constraintWidth_min Sets the minimum width of a view	app:layout_constraintWidth_min="50dp"
app:layout_constraintHeight_min Sets the minimum height of a view	app:layout_constraintHeight_min="50dp"

Advantages of Constraint Layout

Flexibility: Allows for complex positioning and alignment of views

Performance: Maintains a flat view hierarchy, which can improve performance

Responsiveness: Easily adapts to different screen sizes and orientations

Pavithra.S

Ease of Use: The visual editor in Android Studio makes it easy to create and manage constraints

Disadvantages of Constraint Layout

Learning Curve: Can be complex and difficult to learn for beginners

Overhead: Might add unnecessary complexity for simple layouts

Performance: While it can improve performance in some cases, improper use of constraints can lead to performance issues

17. Explain the steps in using Custom Content Provider

18. Provide a detailed explanation of how to integrate Google Maps into an Android application