



# **BACHELOR OF COMPUTER APPLICATIONS**

## **SEMESTER 6**

### **DCA3201**

# **MOBILE APPLICATION DEVELOPMENT**

# Unit 2

## Android Development

### Table of Contents

SL No	Topic	Fig No / Table / Graph	SAQ / Activity	Page No
1	<a href="#">Introduction</a>	-	-	4
1.1	<a href="#">Learning Objectives</a>	-	-	
2	<a href="#">System Requirements for Android Development</a>	-	-	5 - 7
2.1	<a href="#">Introduction</a>	-	-	
2.2	<a href="#">Understanding the Importance of System Specifications</a>	-	-	
2.3	<a href="#">Navigating Between Minimum and Optimal System Requirements</a>	-	-	
2.4	<a href="#">Baseline System Prerequisites</a>	-	-	
2.5	<a href="#">Optimal System Recommendations</a>	-	-	
2.6	<a href="#">Special Considerations for Emulator Usage</a>	-	-	
2.7	<a href="#">Additional Development Tools and Software</a>			
2.8	<a href="#">Evaluating System Compatibility</a>			
3	<a href="#">Installing and Configuring Android SDK</a>	-	-	8 - 11
3.1	<a href="#">Why Install the Android SDK?</a>	-	-	
3.2	<a href="#">Prerequisites</a>	-	-	
3.3	<a href="#">Installation Process</a>	-	-	
3.4	<a href="#">Configuring the Android SDK</a>	-	-	
3.5	<a href="#">Using the Android SDK Manager</a>	-	-	
3.6	<a href="#">Emulator &amp; Virtual Devices</a>			

	3.7	<a href="#">Advanced Configuration</a>			
4		<a href="#">Introduction to Java: Basic Concepts and Syntax</a>	-	-	12 - 20
	4.1	<a href="#">A Glimpse into Java's Genesis</a>	-	-	
	4.2	<a href="#">Why Java?</a>	-	-	
	4.3	<a href="#">Basic Concepts</a>	-	-	
5		<a href="#">Setting Up Android Studio</a>	-	-	21 - 36
	5.1	<a href="#">System Requirements</a>	-	-	
	5.2	<a href="#">Downloading Android Studio</a>	-	-	
	5.3	<a href="#">Installation Process</a>	-	-	
	5.4	<a href="#">Configuring Android Studio</a>	-	-	
	5.5	<a href="#">Step-by-step installation of Android studio on Windows</a>	-	-	
6		<a href="#">Summary</a>	-	-	37
7		<a href="#">Self- Assessment Questions</a>	-	-	38 - 40
8		<a href="#">Terminal Questions</a>	-	-	41
9		<a href="#">Glossary</a>	-	-	42 - 43
10		<a href="#">Answers</a>	-	-	44
11		<a href="#">References</a>	-	-	45

## 1. INTRODUCTION

Delving into the world of Android development, it's essential to establish a robust and efficient development environment to support your endeavours. Such an environment hinges on having the right system requirements and tools. From understanding the importance of system specifications to navigating between minimum and optimal requirements, developers need to ensure their workstation is tailored for the task at hand. With baseline prerequisites including specific operating systems, processors, memory, storage, and the Java Development Kit, one can embark on their Android development journey. However, for an enhanced and seamless experience, adhering to optimal system recommendations and considerations, especially if using emulators, is crucial. Additionally, integrating various development tools and software, and evaluating system compatibility, ensures a smooth developmental voyage.

### 1.1 Learning Objectives

*At the end of this unit, students will be able to:*

- ❖ *Describe the importance of meeting specified system requirements.*
- ❖ *Identify the prerequisites for installing the Android SDK.*
- ❖ *List and define Java's basic concepts such as variables, operators, and control structures.*
- ❖ *Install and configure Android Studio on a personal system*

## 2. SYSTEM REQUIREMENTS FOR ANDROID DEVELOPMENT

### 2.1 Introduction

Delving into Android development requires not only passion but also a robust and efficient development environment. This environment, encompassing the Android Software Development Kit (SDK) and associated tools, serves as the backbone of your developmental endeavours. Hence, aligning your computer system with the optimal specifications is paramount.

### 2.2 Understanding the Importance of System Specifications

Your development workstation is analogous to a craftsman's workshop. Just as precision tools are vital for impeccable craftsmanship, a developer's efficiency hinges on a system tailored for the task. Suboptimal setups can introduce bottlenecks, causing extended compile times, emulator latency, and potential system instabilities. Thus, ensuring your system's adequacy is of utmost importance.

### 2.3 Navigating Between Minimum and Optimal System Requirements

While adhering to the basic system prerequisites enables Android development initiation, for a fluid and streamlined experience, it's prudent to align with the optimal specifications.

### 2.4 Baseline System Prerequisites

- Operating System: Windows 7/8/10, macOS, or a pertinent Linux distribution.
- Processor: Intel i3 or an equivalent variant.
- Memory: 4GB RAM.
- Storage: Approximately 2GB for the Android SDK and emulator system images.
- Java Development Kit (JDK): Version 8 or subsequent.

### 2.5 Optimal System Recommendations

- Operating System: Windows 10, macOS Catalina, or a contemporary Linux variant.
- Processor: Intel i5/i7 or a comparable high-caliber variant.



- Memory: Preferably 8GB RAM or above.
- Storage: Above 4GB to facilitate the Android SDK, emulator system images, and additional buffer.
- Java Development Kit (JDK): The latest version for enhanced features and performance.

## 2.6 Special Considerations for Emulator Usage

Emulators, which offer a virtual representation of Android devices, can be notably resource intensive. For developers relying extensively on emulators, the following specifications are vital:

- Hardware Virtualization: Ensure compatibility with Intel Virtualization Technology (VT-x) or AMD-V and activation in the BIOS settings.
- Graphics Processing Unit (GPU): Incorporating a dedicated GPU can significantly augment emulator efficacy.
- Memory Allocation: Dedicate an additional 2GB of RAM specifically for the emulator, beyond the system's primary allocation.

## 2.7 Additional Development Tools and Software

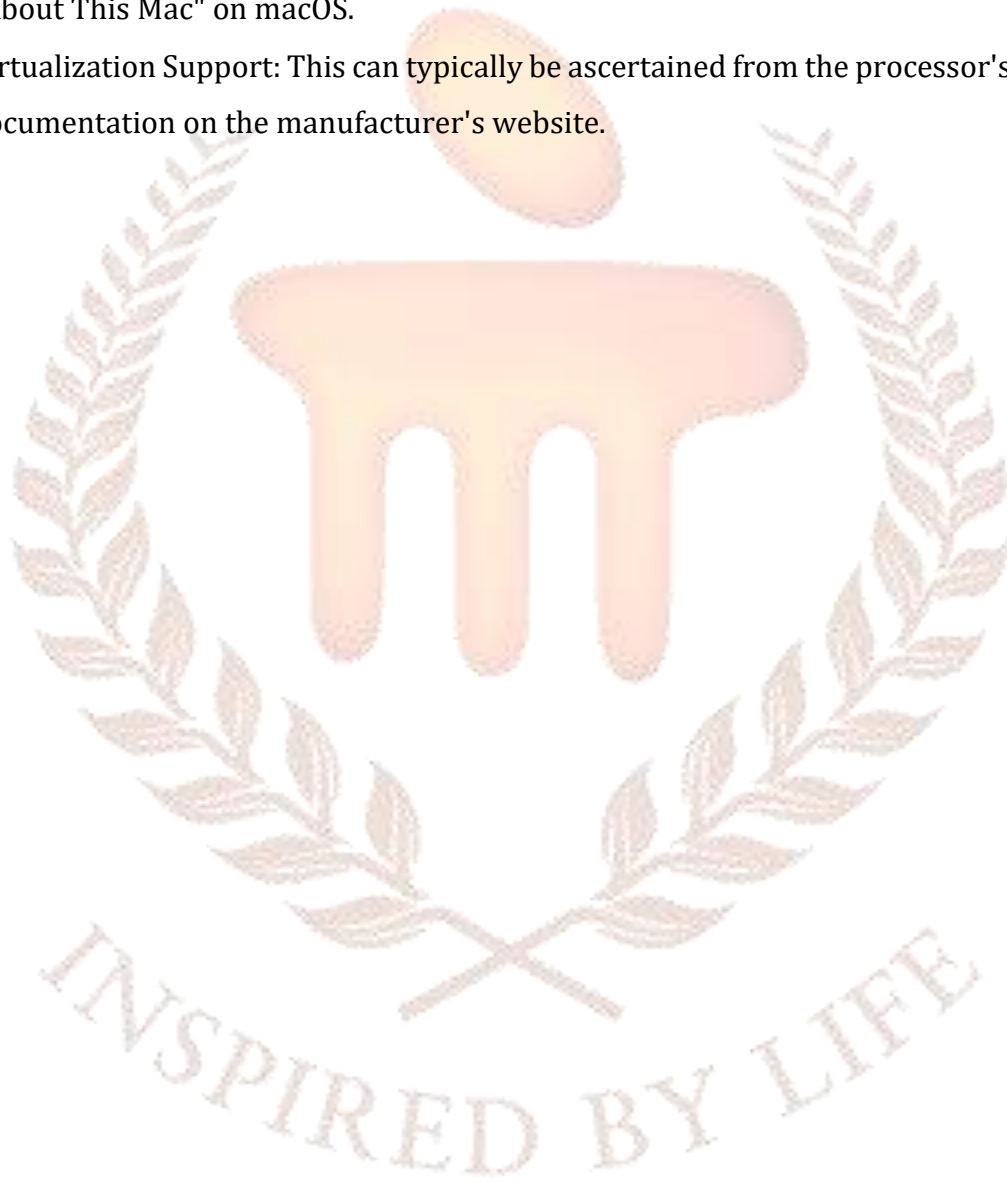
In conjunction with the primary system requirements, the development toolkit often encompasses:

- Version Control Systems: Tools such as Git, facilitating collaborative development and version management.
- Integrated Development Environment (IDE): Android Studio is the official IDE for Android, though developers might also consider alternatives like Eclipse.
- Web Browsers: Essential for accessing up-to-date documentation, web application testing, and potential troubleshooting.

## 2.8 Evaluating System Compatibility

Prior to configuring the development environment, a meticulous system compatibility assessment is recommended:

- Processor and Memory Evaluation: For Windows users, the "System" properties section is informative. For macOS users, the "About This Mac" option provides the requisite details.
- Storage Assessment: Ensure sufficient storage via "My Computer" on Windows or "About This Mac" on macOS.
- Virtualization Support: This can typically be ascertained from the processor's technical documentation on the manufacturer's website.



### 3. INSTALLING AND CONFIGURING ANDROID SDK

The Android Software Development Kit (SDK) is a comprehensive set of development tools that enables developers to create applications for the Android platform. Whether you're a seasoned developer or a newbie, understanding how to properly install and configure the Android SDK is crucial for a smooth development experience.

#### 3.1 Why Install the Android SDK?

Even if you're using an Integrated Development Environment (IDE) like Android Studio, which bundles the Android SDK by default, there may be times when you need to install the SDK separately. This could be for:

- Developing with other IDEs or tools.
- Using the latest tools that haven't been bundled with Android Studio yet.
- Running specific command-line utilities.

#### 3.2 Prerequisites

Before diving into the installation process, ensure your system meets these prerequisites:

- Operating System: Windows, Mac, or Linux.
- Java Development Kit (JDK): The Android SDK requires the JDK. Make sure to install JDK 8 or newer.

#### 3.3 Installation Process

##### Step 1: Downloading the Android SDK

- Navigate to the official Android developer site: <https://developer.android.com/>
- Look for the standalone SDK download. It's important to note that if you've installed Android Studio, you've already got the SDK, but this guide focuses on the standalone version.



**Step 2: Unpack the Archive**

- Once the download is complete, extract the zip file to an appropriate location on your machine. Remember the path, as you'll need to reference the SDK's location in future development tasks.

**Step 3: Launching the SDK Manager**

- Navigate to the location where you extracted the SDK. Within, you'll find a file named "SDK Manager". On Windows, it's "SDK Manager.exe".
- Run the SDK Manager. This tool allows you to download essential components and tools associated with the Android SDK.

**Step 4: Installing Packages**

- Upon launching the SDK Manager, you'll be presented with a list of tools, platforms, and other components available for download.
- At a minimum, you'll want to download:
  - The latest Android SDK tools.
  - Android platform-tools.
  - The SDK platform for the latest version of Android.
  - Any additional components or libraries you foresee needing.

**3.4 Configuring The Android SDK**

Setting the PATH:

For convenience, add the path to the Android SDK's 'tools' and 'platform-tools' directories to your system's PATH:

- Windows:
  1. Right-click on 'My Computer' and click 'Properties'.
  2. Navigate to 'Advanced System Settings'.
  3. Click on the 'Environment Variables' button.

4. In the 'System Variables' section, find the 'Path' variable, and add the path to your SDK's tools and platform-tools.

- Mac/Linux:
  1. Edit your shell profile file (like '.bashrc' or '.bash\_profile').
  2. Add 'export PATH=\$PATH:/path/to/sdk/tools:/path/to/sdk/platform-tools'.

### 3.5 Using The Android SDK Manager

The SDK Manager is a powerful tool that allows you to manage and update the SDK components. Some essential features include:

- Updating Tools: Regularly check for updates, as Google frequently releases new versions with bug fixes, new features, and other improvements.
- Installing Specific Android Versions: For testing purposes, you might need to download system images for various Android versions.
- Extras: This section in the SDK Manager contains additional tools, libraries, and plugins that can be beneficial for specific development needs.

### 3.6 Emulator & Virtual Devices

One of the standout features of the Android SDK is the Android Emulator. The emulator lets you test your applications on various virtual devices without needing physical hardware.

- AVD Manager: This tool helps you create and manage virtual devices. Each virtual device emulates a specific Android device model, screen size, and Android version.
- Configuring an AVD: When setting up a new AVD, select the desired system image (Android version), adjust hardware options, and define device orientation, scale, and other properties.

### 3.7 Advanced Configuration

For those looking to delve deeper:

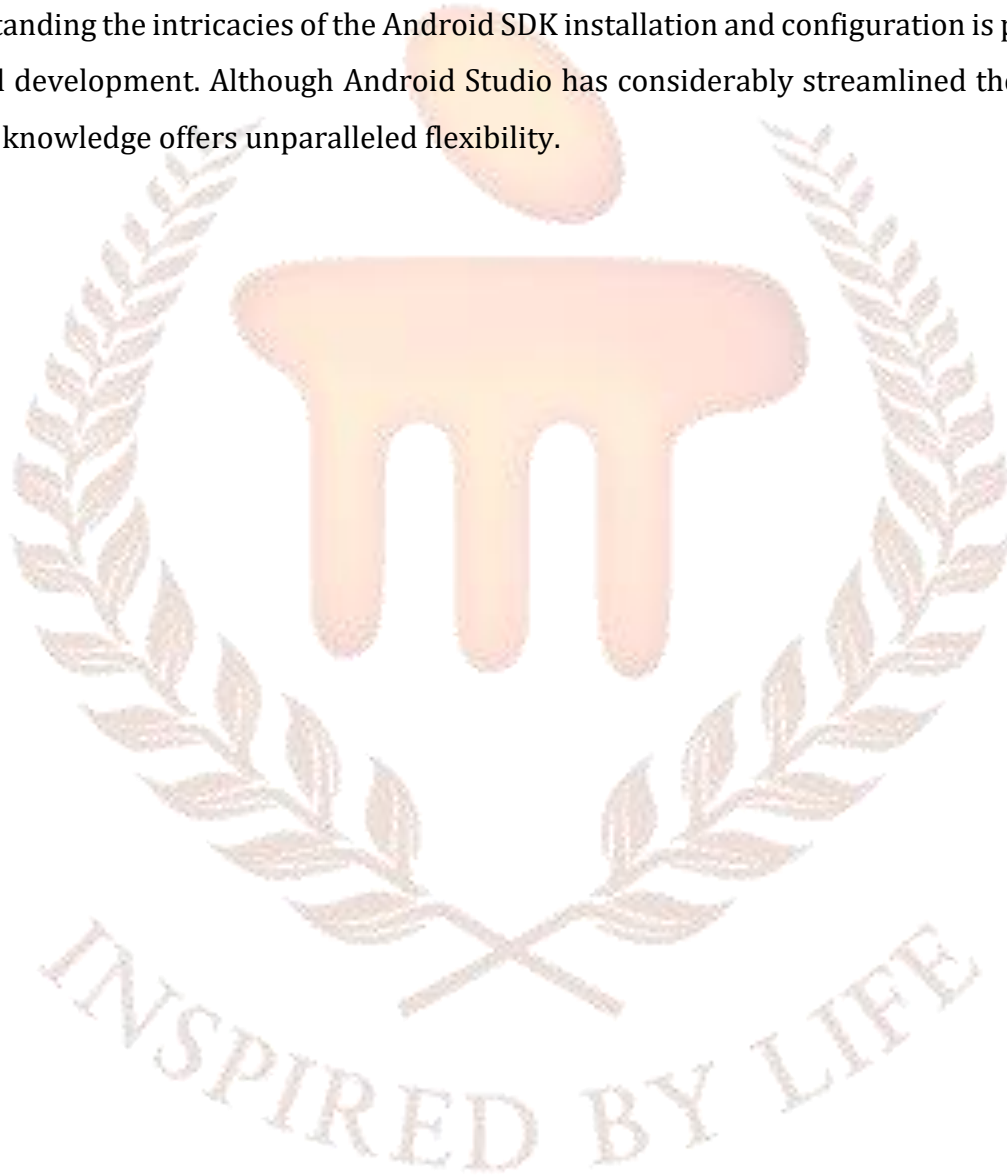
Proxy Settings: If you're behind a proxy, configure the SDK Manager's proxy settings by going to "Tools" -> "Options".

SDK Platforms: Download older versions of Android for compatibility testing.

SDK Update Sites: By default, the SDK Manager fetches from Google's repositories.

However, you can add third-party repositories or remove default ones.

Understanding the intricacies of the Android SDK installation and configuration is pivotal for Android development. Although Android Studio has considerably streamlined the process, manual knowledge offers unparalleled flexibility.



## 4. INTRODUCTION TO JAVA: BASIC CONCEPTS AND SYNTAX

### 4.1 A Glimpse into Java's Genesis

Java, conceived by Sun Microsystems in the early 1990s, is an object-oriented, class-based language pivotal to modern software development. Its "Write Once, Run Anywhere" credo, enabled by its platform independence through the Java Virtual Machine (JVM), revolutionized programming.

### 4.2 Why Java?

Java's popularity is anchored in its:

- Platform Independence: Java applications can run on any device equipped with a JVM.
- Object-Oriented: Facilitates modular and organized coding.
- Multithreading Capabilities: Enables concurrent execution for efficient processing.
- Rich Standard Library: Offers a plethora of pre-compiled classes and methods.

### 4.3 Basic Concepts

#### 4.3.1 Variables

Variables are like containers that store data. In Java, every variable has a data type that determines the kind of data it can store.

#### 4.3.2 Primitive Data Types:

'byte': 8-bit signed integer. Example: `byte a = 65;`

'short': 16-bit signed integer. Example: `short b = 32000;`

'int': 32-bit signed integer. Example: `int c = 100000;`

'long': 64-bit signed integer. Example: `long d = 100000L;`

'float': 32-bit floating point. Example: `float e = 3.14f;`

'double': 64-bit floating point. Example: `double f = 3.1415926535;`

'char': 16-bit Unicode character. Example: `char g = 'A';`

'boolean': Two possible values, true or false. Example: `boolean h = true;`

### 4.3.3 Reference Data Types:

Unlike primitive data types, reference types store references to the memory location where data is kept, not the data itself. The most common examples are objects and arrays.

Example:

`String name = "John";`

### 4.3.4 Operators

Operators in Java are symbols that perform operations on variables and values.

#### 4.3.4.1 Arithmetic Operators:

Addition (+): `int sum = 5 + 3; // Output: sum = 8`

Subtraction (-): `int diff = 5 - 3; // Output: diff = 2`

Multiplication (\*): `int product = 5 * 3; // Output: product = 15`

Division (/): `int quotient = 5 / 3; // Output: quotient = 1`

Modulus (%): `int remainder = 5 % 3; // Output: remainder = 2`

#### 4.3.4.2 Relational Operators:

They determine the relationship between two values.

Equal to (==): `(5 == 3) // Output: false`

Not equal to (!=): `(5 != 3) // Output: true`

Greater than (>): `(5 > 3) // Output: true`

Less than (<): `(5 < 3) // Output: false`

Greater than or equal to ( $\geq$ ): `(5 >= 3) // Output: true`

Less than or equal to ( $\leq$ ): `(5 <= 3) // Output: false`

#### 4.3.4.3 Logical Operators:

They are used to determine the logic between two or more values.

Logical and ( $\&\&$ ): `(5 > 3 &\& 5 > 4) // Output: true`

Logical or ( $\|\|$ ): `(5 > 3 \|\| 5 < 4) // Output: true`

Logical not (!): `!(5 < 3) // Output: true`

#### 4.3.5 Control Structures

Control structures help you control the flow of execution in a program.

##### 4.3.5.1 Decision-making Structures:

‘if’:

```
if (5 > 3) {  
    System.out.println("True");  
}
```

‘if-else’:

```
if (5 < 3) {  
    System.out.println("True");  
} else {  
    System.out.println("False");  
}
```

‘switch’:



```
int day = 3;
switch (day) {
    case 1:
        System.out.println("Monday");
        break;
    case 2:
        System.out.println("Tuesday");
        break;
    case 3:
        System.out.println("Wednesday");
        break;
    default:
        System.out.println("Another day");
}
```

#### 4.3.5.2 Looping Structures:

These structures allow us to execute a statement or a block of statements multiple times.

‘for’:

```
for (int i = 0; i < 3; i++) {
    System.out.println(i);
}
```

‘while’:

```
int i = 0;
while (i < 3) {
    System.out.println(i);
    i++;
}
```

```
int i = 0;
do {
    System.out.println(i);
    i++;
} while (i < 3);
```

### 4.3.6 Dive into Java Syntax

#### 4.3.6.1 Java Program Structure

A basic Java program consists of:

```
package mypackage;

import java.util.*;

public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
```

#### 4.3.6.2 Comments in Java

Comments are used to explain the code and are not executed.

- Single-line:

```
// This is a single-line comment
```

- Multi-line:

```
/*
This is a
multi-line comment
*/
```

### 4.3.6.3 Naming Conventions

Java naming conventions make programs more readable.

- **Classes:** They should start with an uppercase letter and be nouns. E.g., 'Bicycle', 'Car'.
- **Methods:** They should start with a lowercase letter and be verbs. E.g., 'calculateSpeed()', 'setDetails()'.

### 4.3.6.4 Classes and Objects

In Java, the concept of OOP (Object-Oriented Programming) revolves around classes and objects.

#### 4.3.6.4.1 Class Definition

A class is a blueprint for creating objects.

```
public class Dog {  
    // attribute  
    String breed;  
  
    // method  
    void bark() {  
        System.out.println("Woof!");  
    }  
}
```

#### 4.3.6.4.2 Object Creation

```
Dog myDog = new Dog();  
myDog.breed = "Beagle";  
myDog.bark();
```

#### 4.3.6.5 Methods in Java

Methods in Java are blocks of code that perform a specific task.

Example:

```
public int addNumbers(int a, int b) {  
    return a + b;  
}
```

#### 4.3.7 Arrays

Arrays in Java are used to store multiple values in a single variable.

Example:

```
int[] numbers = {1, 2, 3, 4, 5};  
System.out.println(numbers[2]);
```

##### 4.3.7.1 Types of Arrays

- **Single Dimensional Arrays:**

These are the simplest form of arrays that enable you to organize a series of items sequentially. This means you can access items by their position in the collection.

Example:

```
int[] ages = {12, 15, 18, 20, 25};  
System.out.println(ages[3]);
```

- **Multidimensional Arrays:**

Java supports arrays with multiple dimensions. The most common form is the two-dimensional array.

Example:

```
int[][] matrix = {  
    {1, 2, 3},  
    {4, 5, 6},  
    {7, 8, 9}  
};  
System.out.println(matrix[1][2]);
```

- Jagged Arrays:

These are arrays that have arrays as their elements. The unique feature of a jagged array is that each row can have a different number of columns.

Example:

```
int[][] jaggedArray = {  
    {1, 2},  
    {3, 4, 5},  
    {6, 7, 8, 9}  
};
```

#### 4.3.7.1 Few Array Operations

Length of an Array:

One can find the length of an array using the length attribute.

Example:

```
int[] numbers = {10, 20, 30, 40, 50};  
System.out.println(numbers.length);
```

### Looping through an Array:

Arrays are often processed with loops. You can iterate over each element of an array using a for loop.

#### Example:

```
int[] numbers = {10, 20, 30, 40, 50};  
System.out.println(numbers.length);
```

### Copying Arrays:

Java provides the `System.arraycopy()` method to copy an array from the specified source array to the destination array.

#### Example:

```
int[] original = {1, 2, 3, 4, 5};  
int[] copied = new int[5];  
System.arraycopy(original, 0, copied, 0, 5);  
for(int num : copied) {  
    System.out.print(num + " ");  
}
```



## 5. SETTING UP ANDROID STUDIO

Android Studio is the official Integrated Development Environment (IDE) for Android application development. It provides a plethora of tools for building apps on every type of Android device. Powered by the IntelliJ IDEA software from JetBrains, Android Studio offers features like code editing, debugging, performance tooling, a flexible build system, and an instant build/deploy system. This guide provides a step-by-step walkthrough to set up Android Studio on your system.

### 5.1 System Requirements

Before delving into the installation process, it's essential to ensure that your system meets the necessary requirements:

For Windows:

- Microsoft Windows 7/8/10 (32- or 64-bit).
- 4 GB RAM minimum, 8 GB RAM recommended.
- 2 GB of available disk space minimum, 4 GB recommended.
- 1280 x 800 minimum screen resolution.

For macOS:

- Mac OS X 10.10 (Yosemite) or higher, up to 10.14 (macOS Mojave).
- 4 GB RAM minimum, 8 GB RAM recommended.
- 2 GB of available disk space minimum, 4 GB recommended.
- 1280 x 800 minimum screen resolution.

For Linux:

- GNOME or KDE desktop.
- 64-bit distribution capable of running 32-bit applications.
- GNU C Library (glibc) 2.19 or later.
- 2 GB of available disk space minimum, 4 GB recommended.
- 1280 x 800 minimum screen resolution.

## 5.2 Downloading Android Studio

- Navigate to the official Android Studio website at <https://developer.android.com/studio>.
- Click on the "Download Android Studio" button.
- Accept the terms and conditions.
- Based on your OS (Windows/macOS/Linux), the appropriate file will start downloading (e.g., .exe for Windows, .dmg for macOS, .tar.gz for Linux).

## 5.3 Installation Process

For Windows:

- Run the .exe file that was downloaded.
- Follow the setup wizard's guidance, which includes downloading the Android SDK components.
- Choose a suitable location for Android Studio and the Android SDK if you wish to change the default.

For macOS:

- Open the .dmg file.
- Drag and drop Android Studio into the Applications folder.
- Launch Android Studio. If you encounter a security warning, navigate to System Preferences -> Security & Privacy and allow the app to run.
- The setup wizard will guide you through the rest, which includes downloading the Android SDK components.

For Linux:

- Unpack the .tar.gz file you downloaded to an appropriate location for your applications.
- Navigate to the android-studio/bin/ directory and execute studio.sh to launch the setup.

- Follow the Android Studio Setup Wizard, which involves downloading the Android SDK components.

## 5.4 Configuring Android Studio

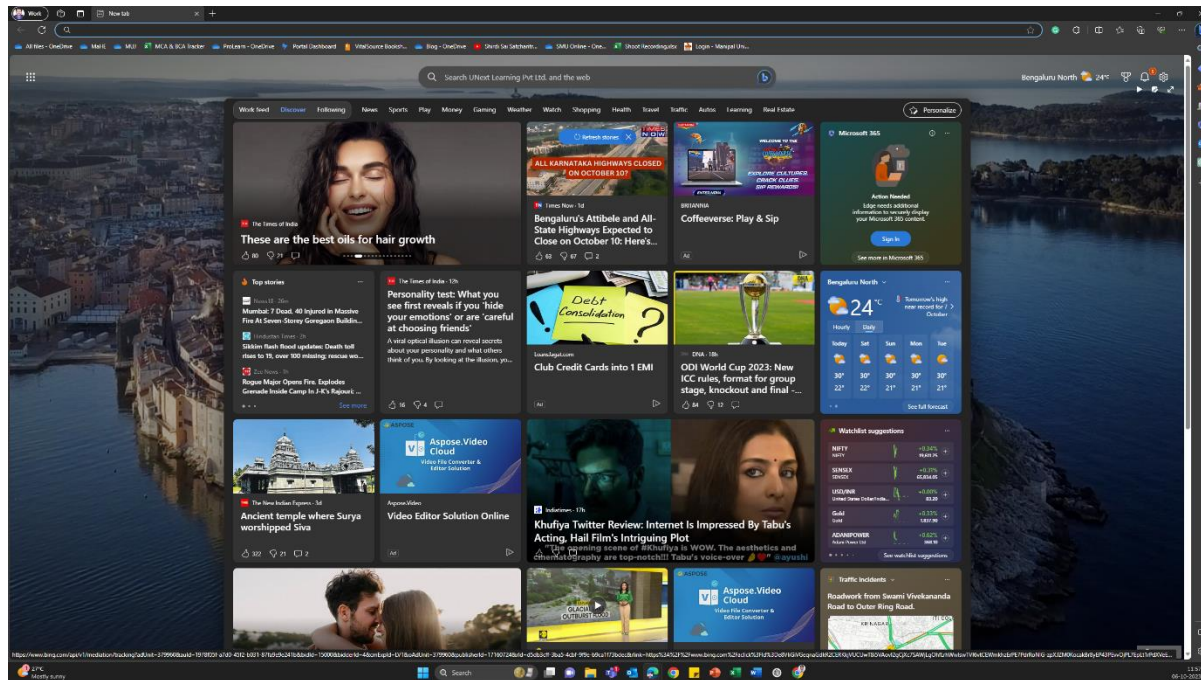
- **Android SDK:** After installation, it's crucial to have the Android Software Development Kit (SDK) correctly set up. The SDK is a collection of APIs, tools, and resources that allow you to develop Android applications. Android Studio typically sets up the latest Android SDK version by default.
- **AVD Manager:** The Android Virtual Device (AVD) Manager allows you to create and manage Android Emulator instances. These emulators enable developers to test apps on various Android devices and versions without needing physical devices.
- **Plugins:** Android Studio supports a myriad of plugins to enhance its capabilities. Navigate to File -> Settings (Preferences on macOS) -> Plugins to search and manage plugins.
- **Themes:** Personalize your Android Studio appearance via themes. The default theme is "IntelliJ," but many developers prefer the "Darcula" theme for a dark mode.

### Important Tips

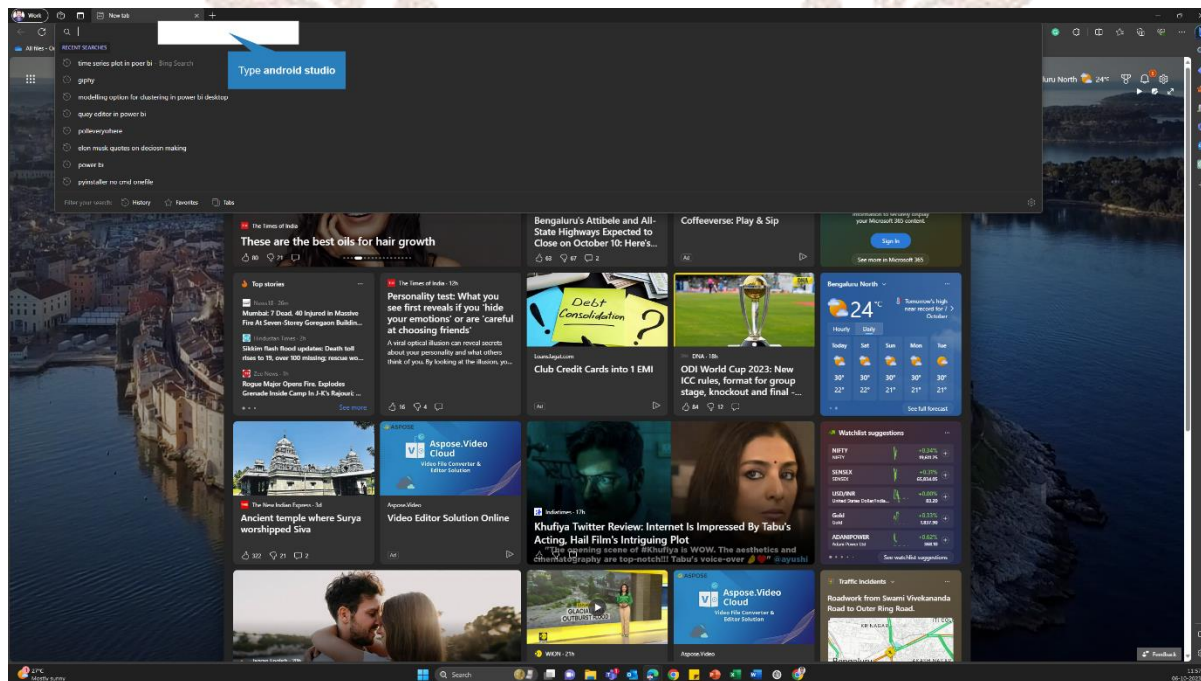
- **Regular Updates:** Android Studio frequently receives updates, including bug fixes, performance improvements, and new features. Regularly updating ensures you have the best environment for Android app development.
- **SDK Platforms:** While Android Studio might install the latest Android version by default, you might need to develop or test on older versions. You can download additional versions via the SDK Manager.
- **Performance:** Running emulators requires a significant amount of system resources. If you're experiencing sluggish performance, consider closing unused applications or upgrading your system's hardware.

## 5.5 Step-by-step installation of Android studio on Windows

Step1: Open the browser on the system.

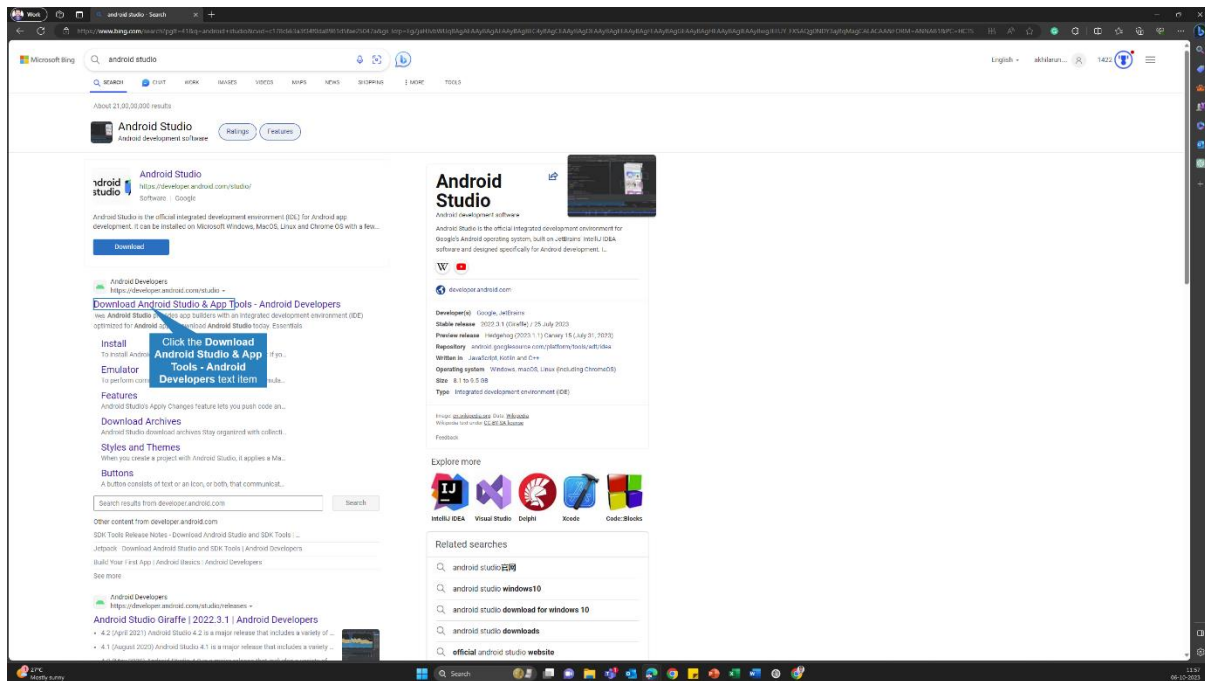


Step 2: On the search bar Type android studio and hit enter.

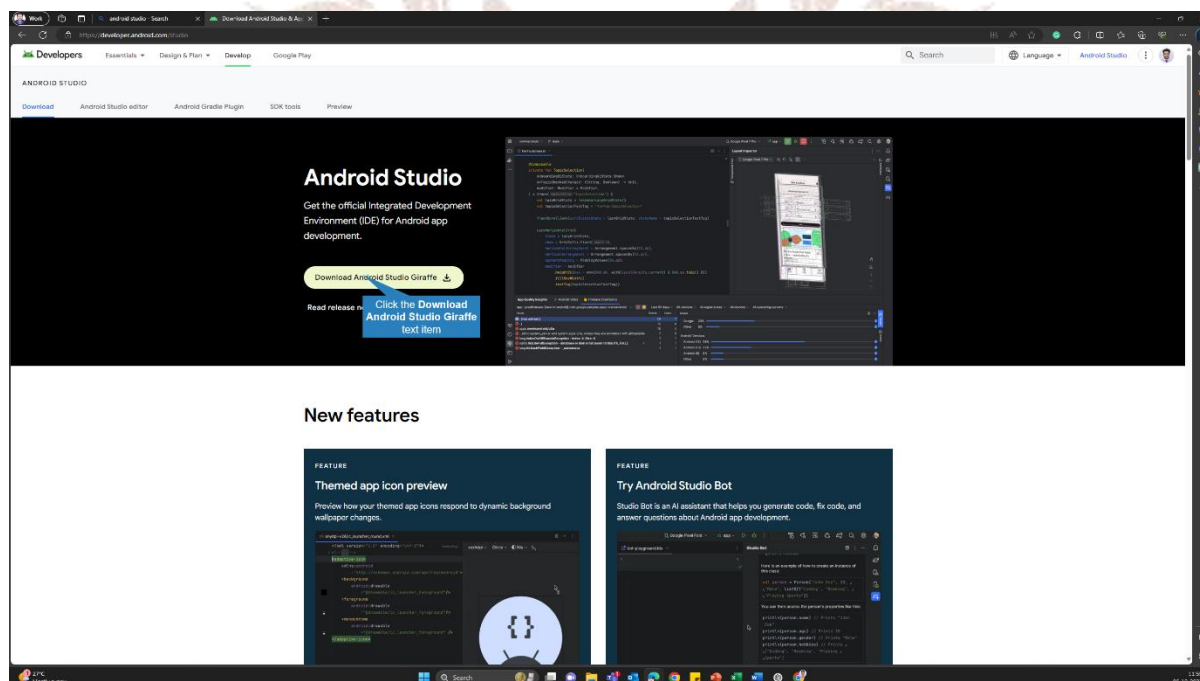




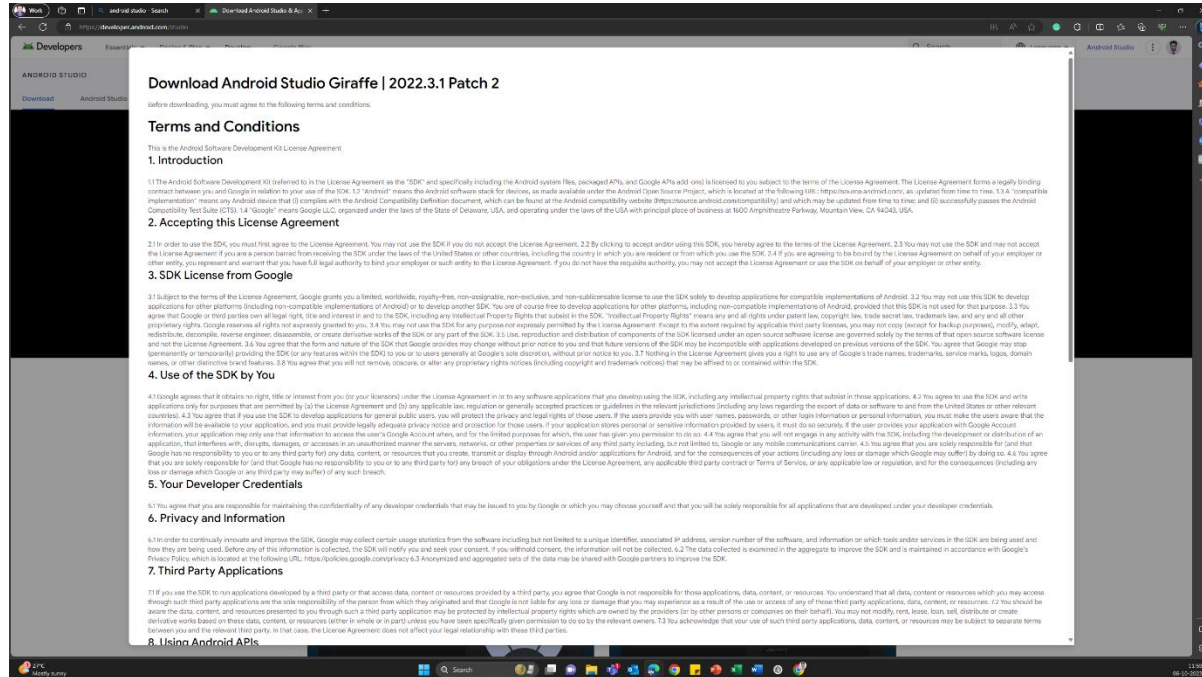
Step 3: In the search list window click the download Android Studio & App Tools - **Android Developers**.



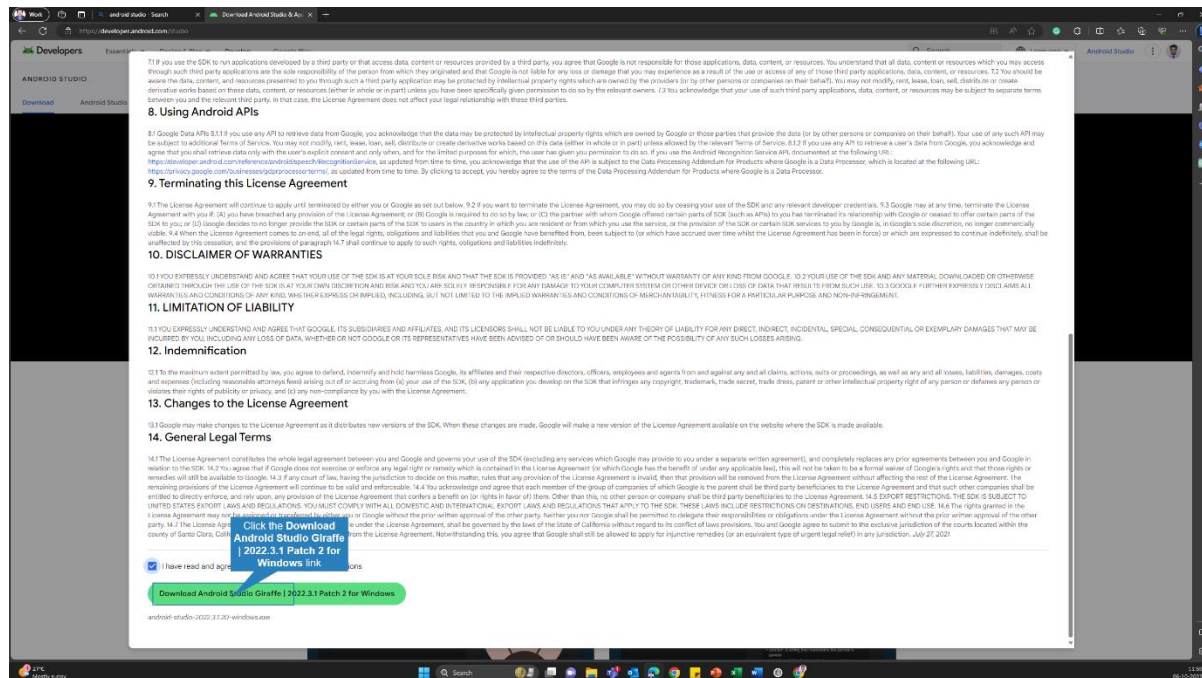
Step 4: On the Android studio website Click the Download Android Studio Giraffe.



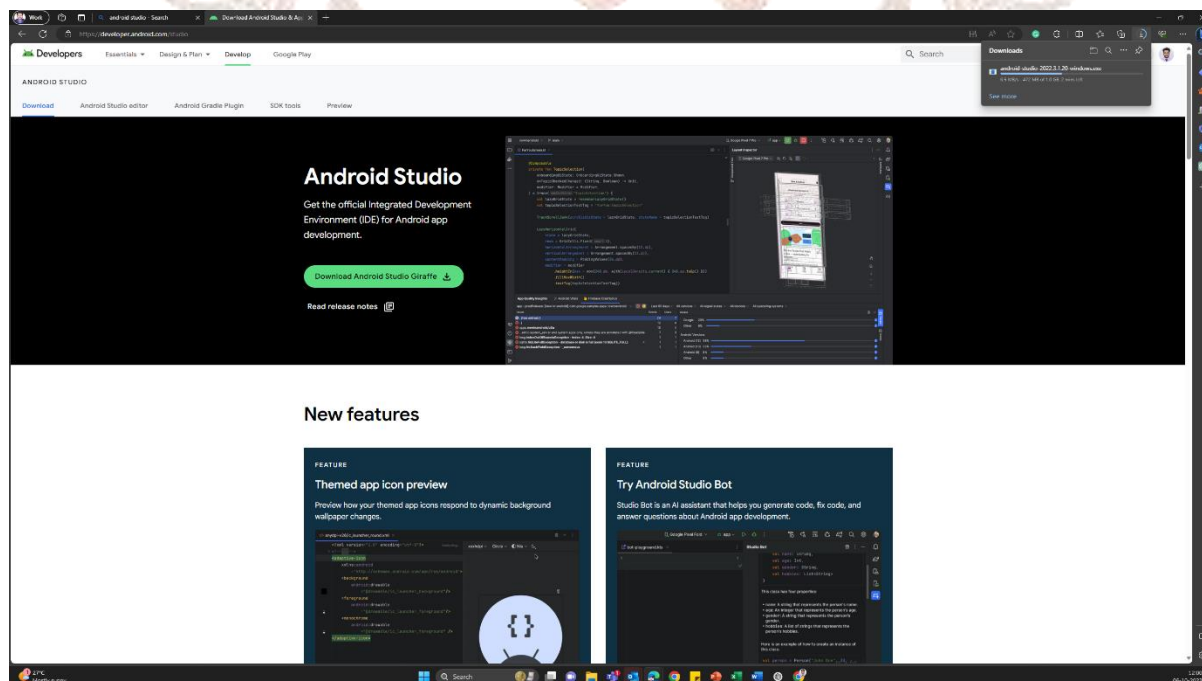
## Step 5: Read the Terms and Conditions carefully and then click the **I have read and agree** with the above terms and conditions checkbox button and click the **Download Android Studio Giraffe | 2022.3.1 Patch 2 for Windows** link.

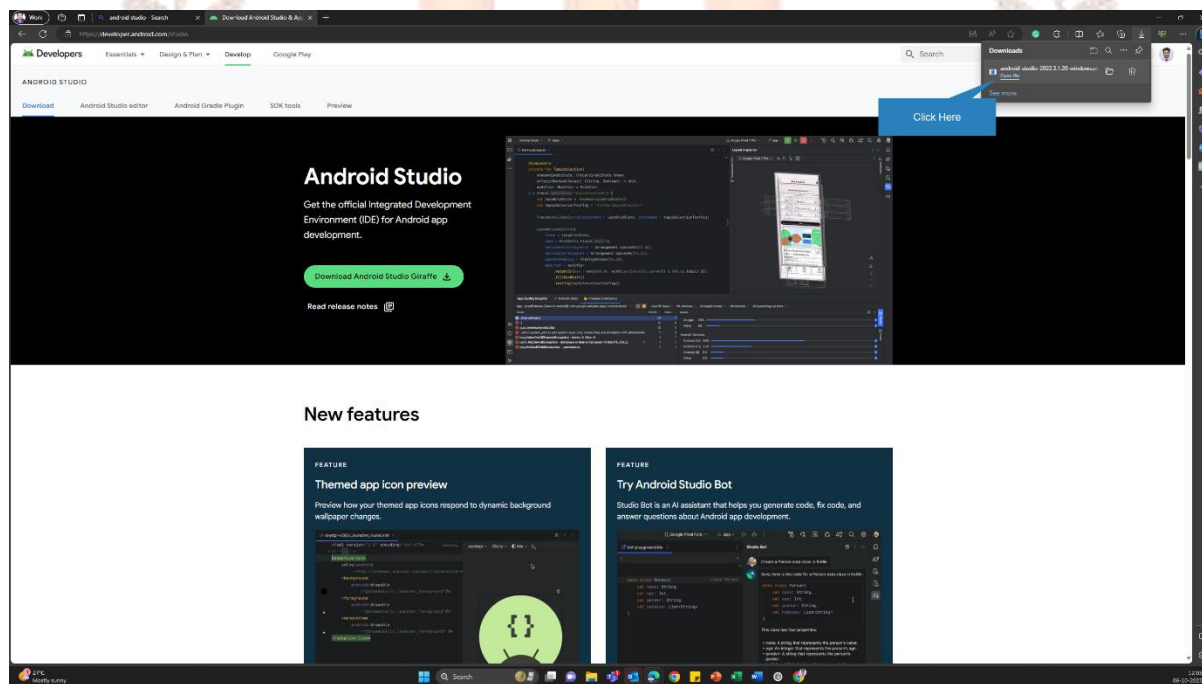
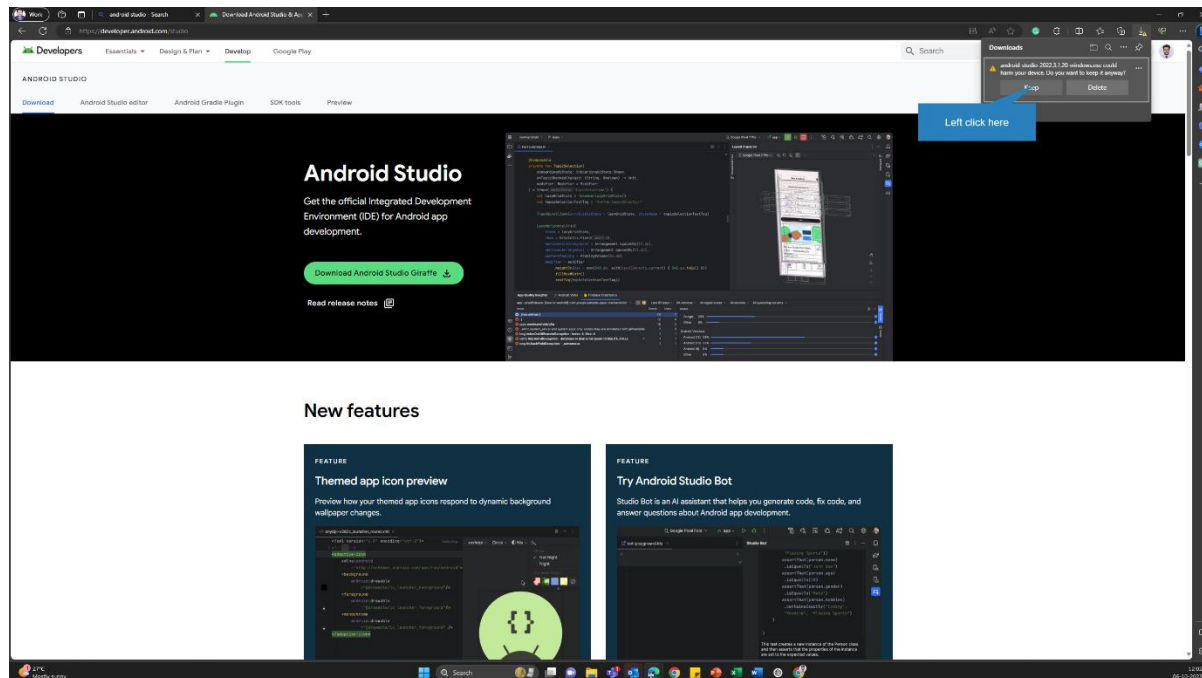


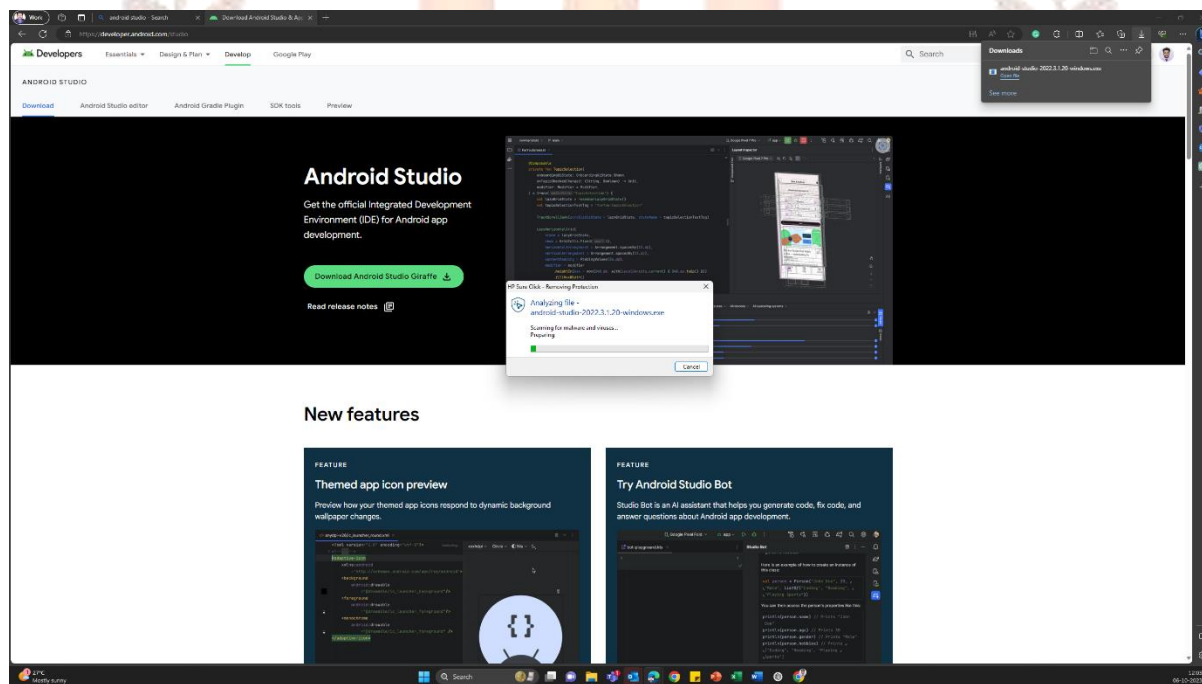
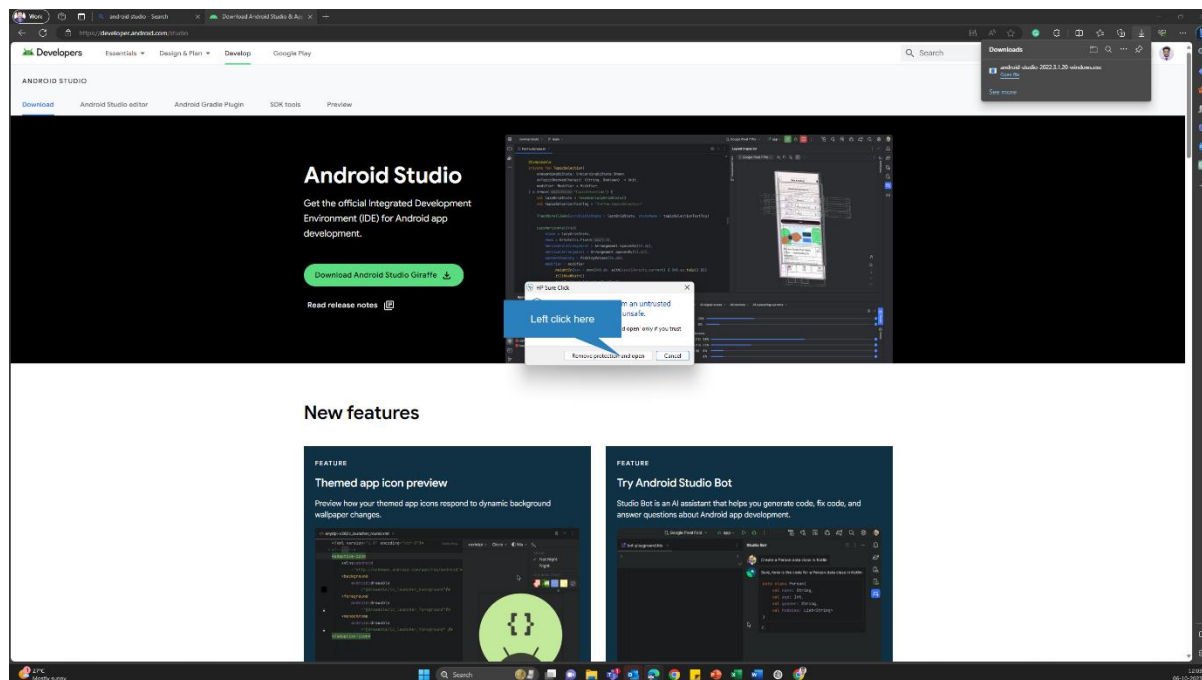




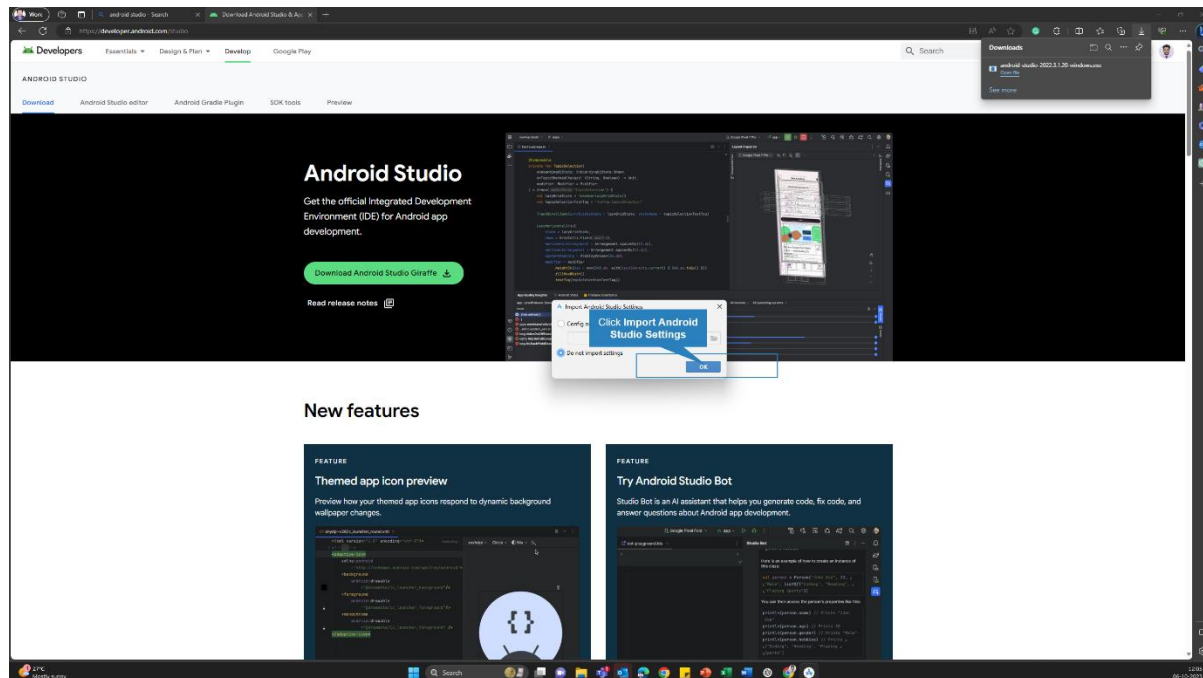
Step 6: Once downloaded click on the open file link to start the installation process and in the pop-up windows click on “Remove protection and open” and the installation process starts.



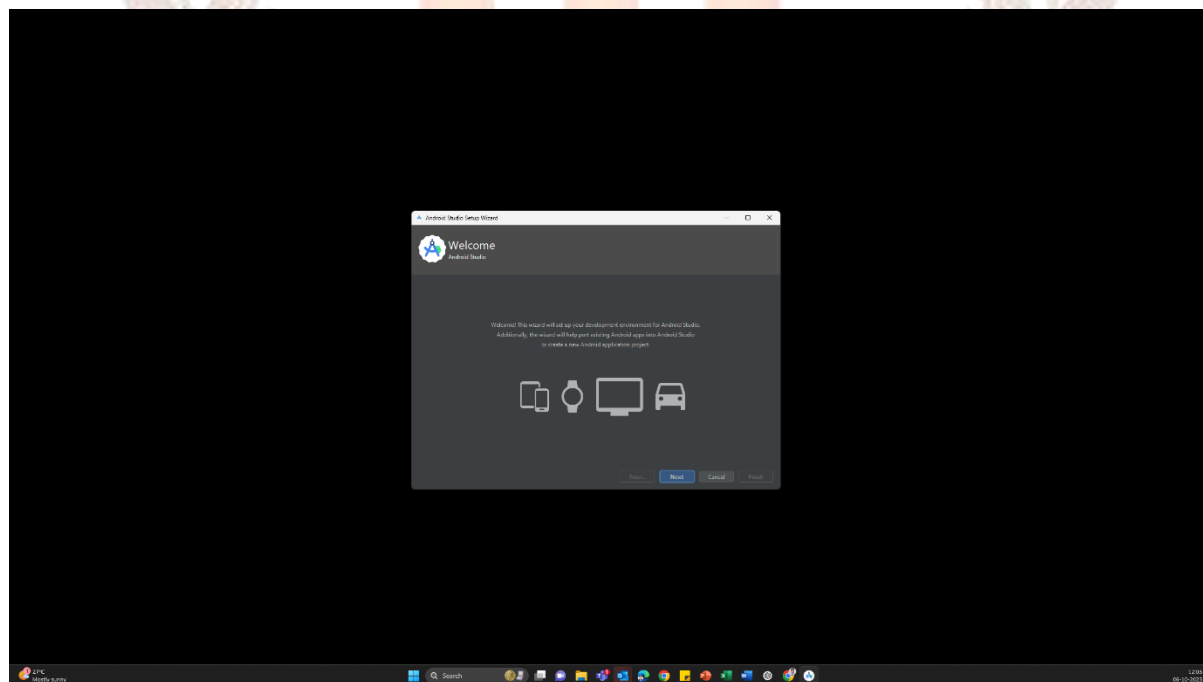




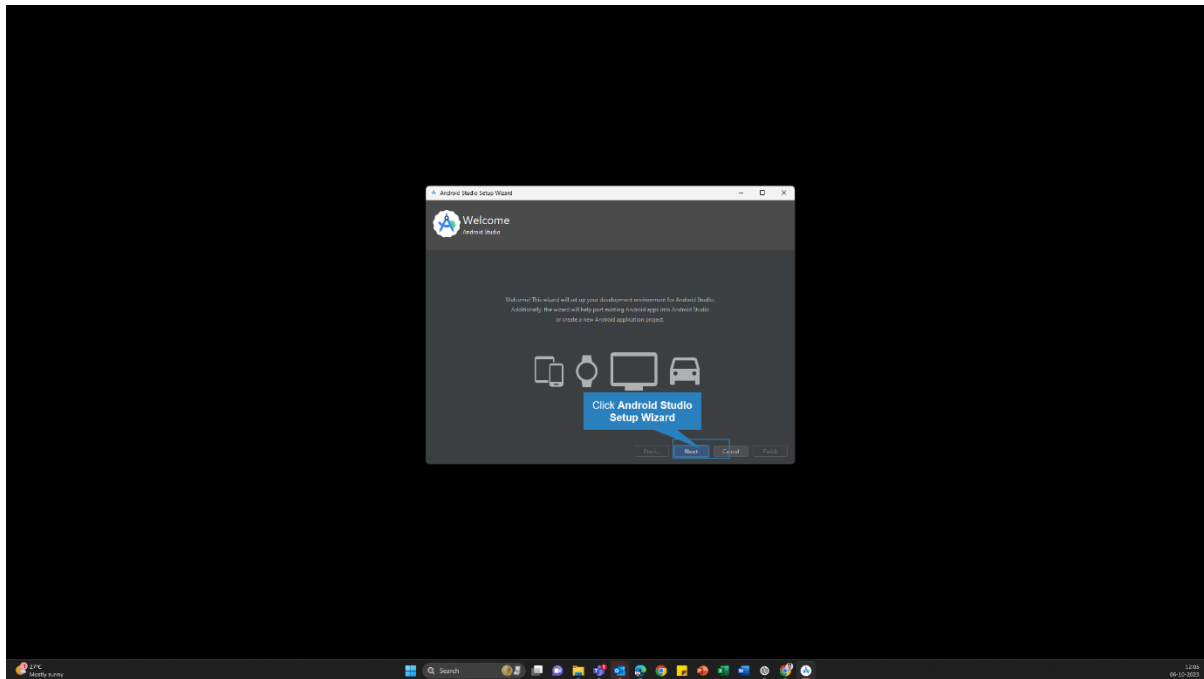
Step 7: During the installation process you will be prompted to select option, click **Import Android Studio Settings** and press ok button.



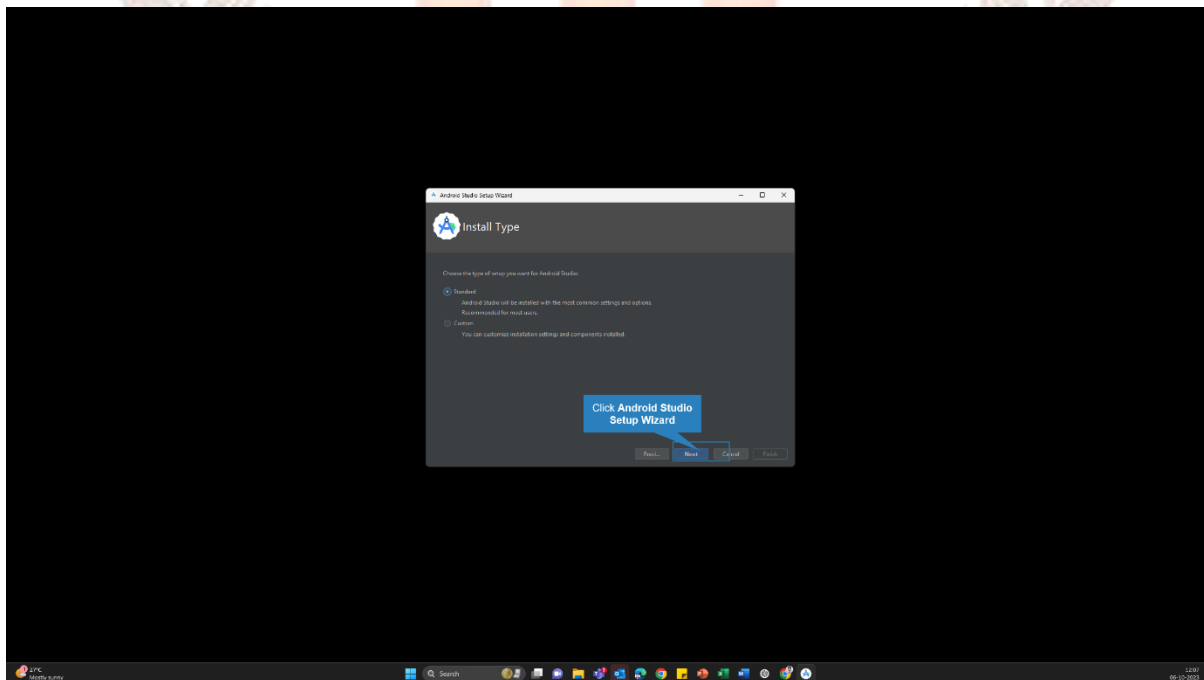
Step 8: Once successfully installed the welcome page gets displayed.



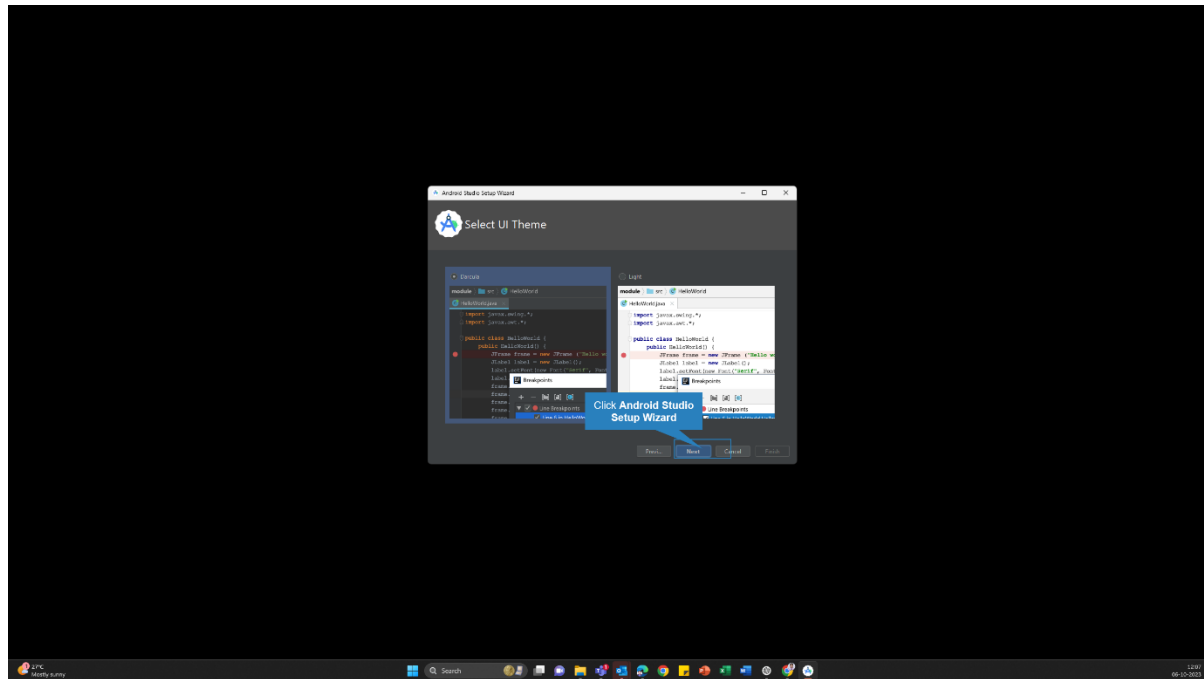
Step 9: In the welcome page dialog box on the Android Studio Setup Wizard click on Next button.



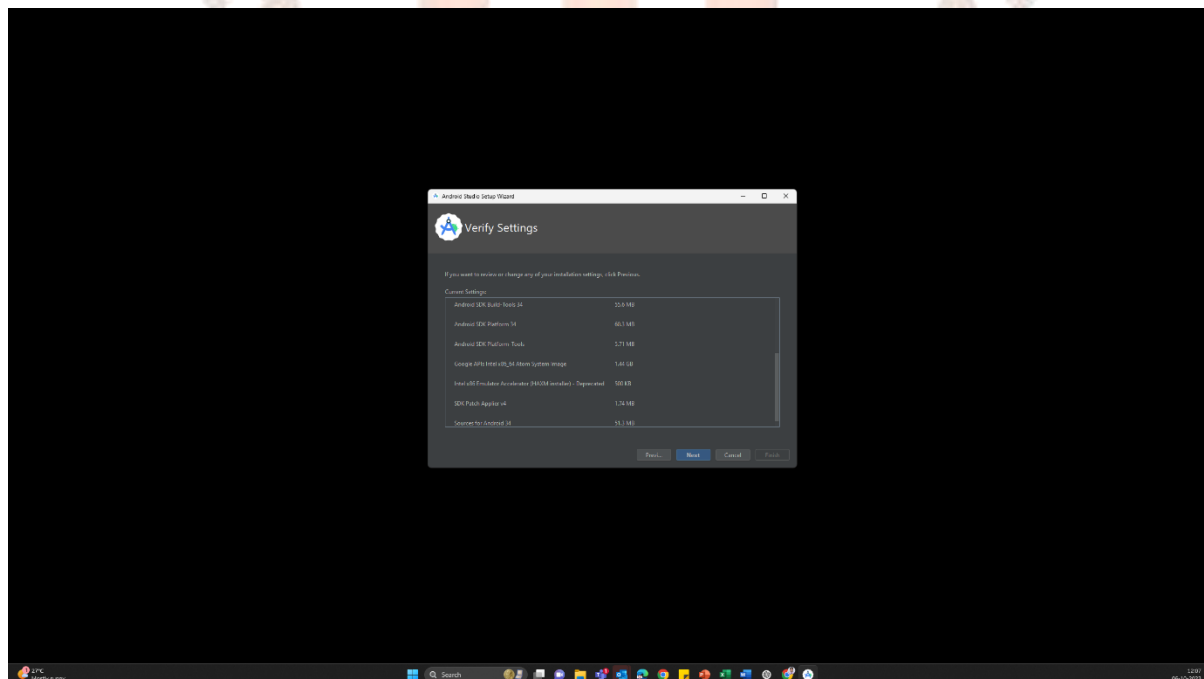
Step 10: On the Install Type dialog box select the Standard type and click on next button.



Step 11: Now, make the selection for the UI theme.

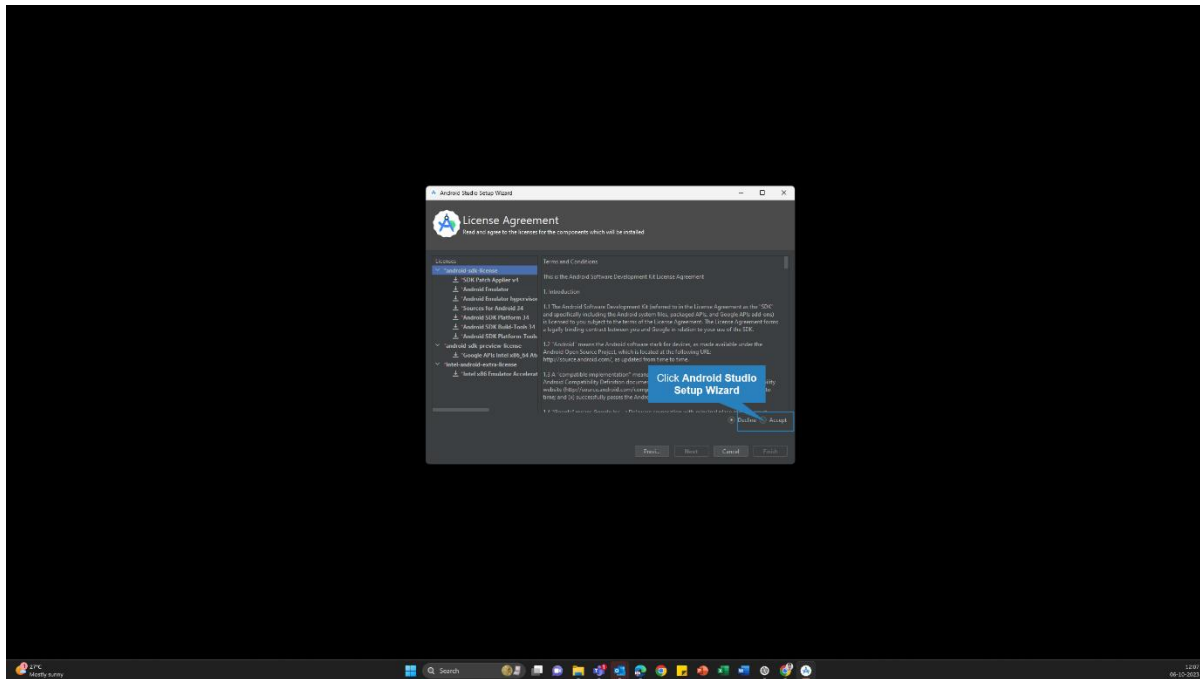


Step 12: Once done with the required step verify the settings done and proceed further by clicking on the next button.



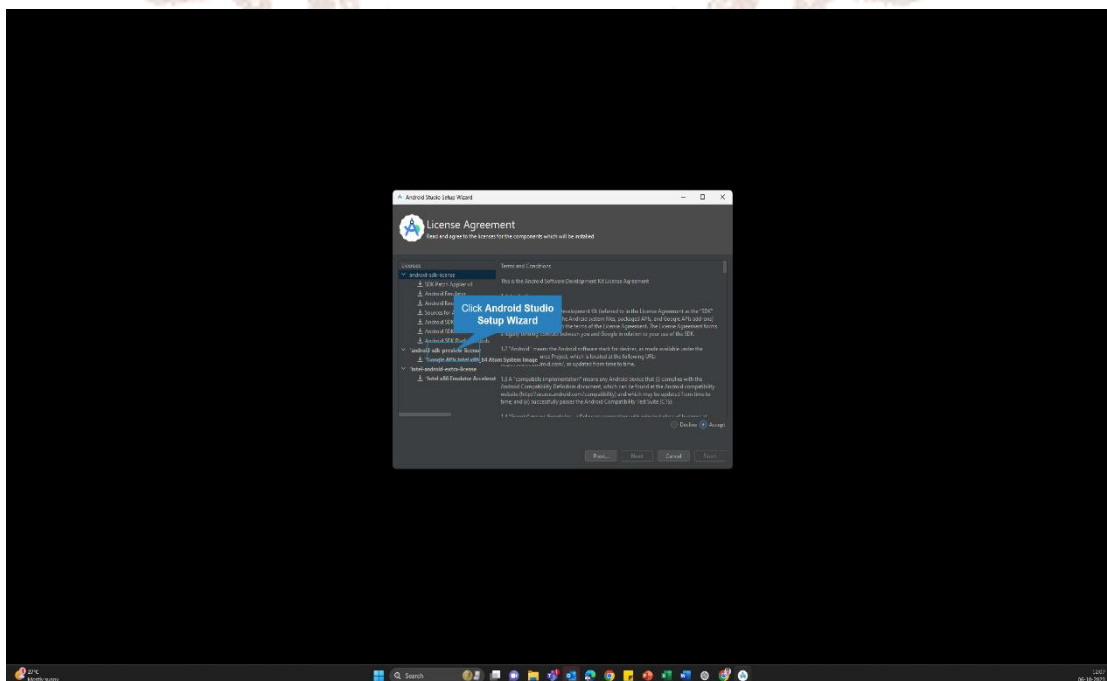
Step 13: In the License Agreement dialog box click on the Android Studio Setup Wizard.



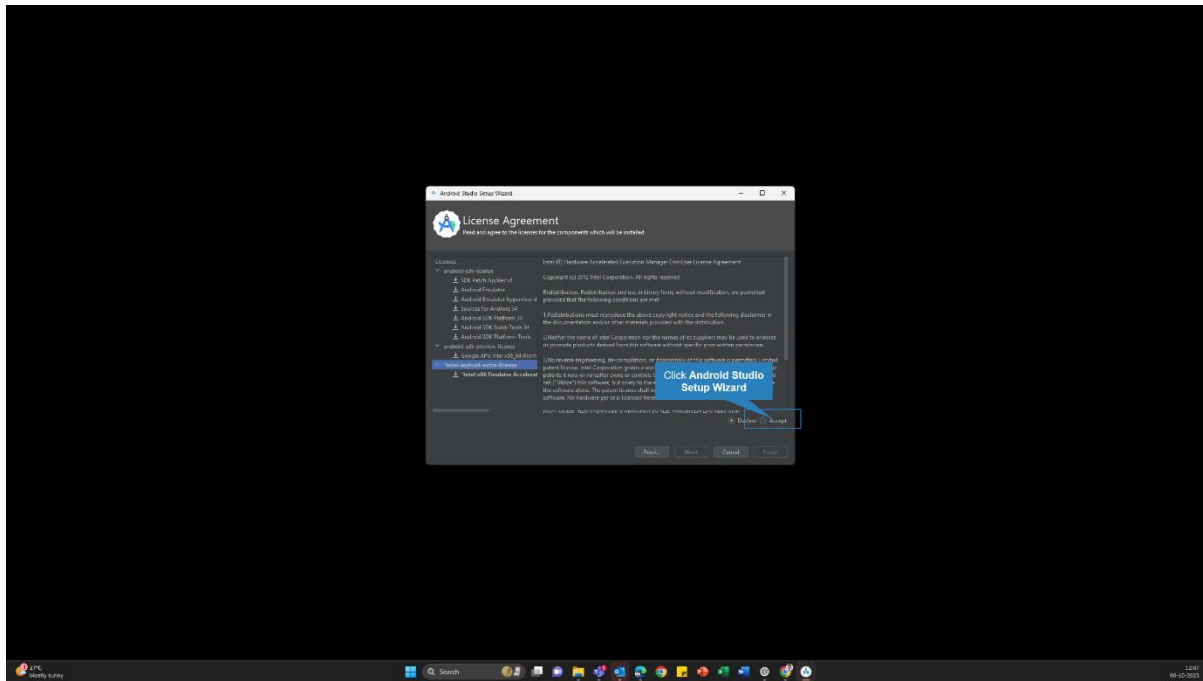


Step 14: In this step select the required licenses under the licenses drop down menu towards the left of the Android Studio setup wizard

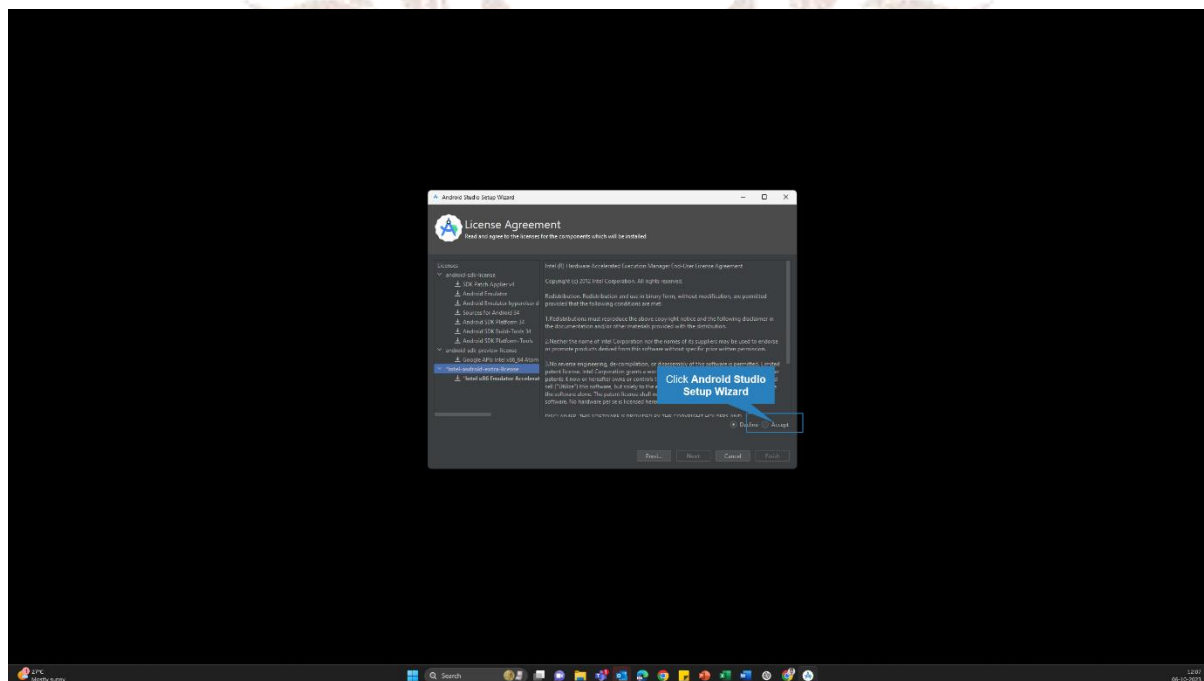
- a. Select adroid-sdk-preview-license and select the accept radio button to accept the License agreement.



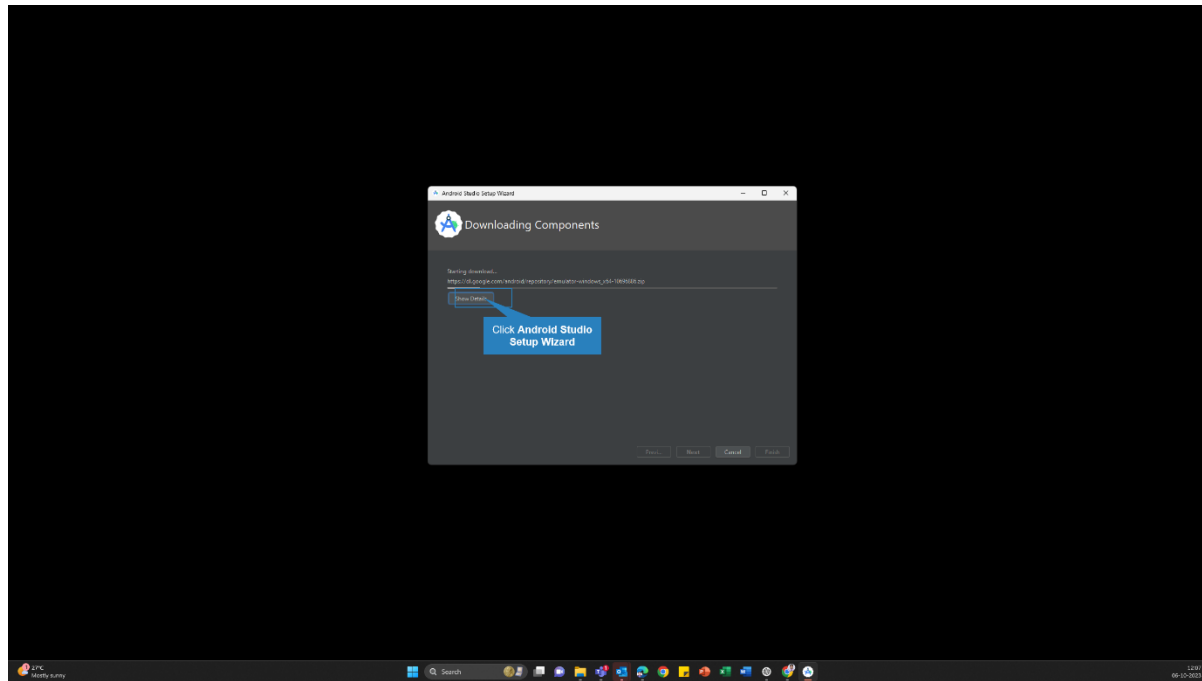
- b. Select intel-android-extra-license and select the accept radio button to accept the License agreement.



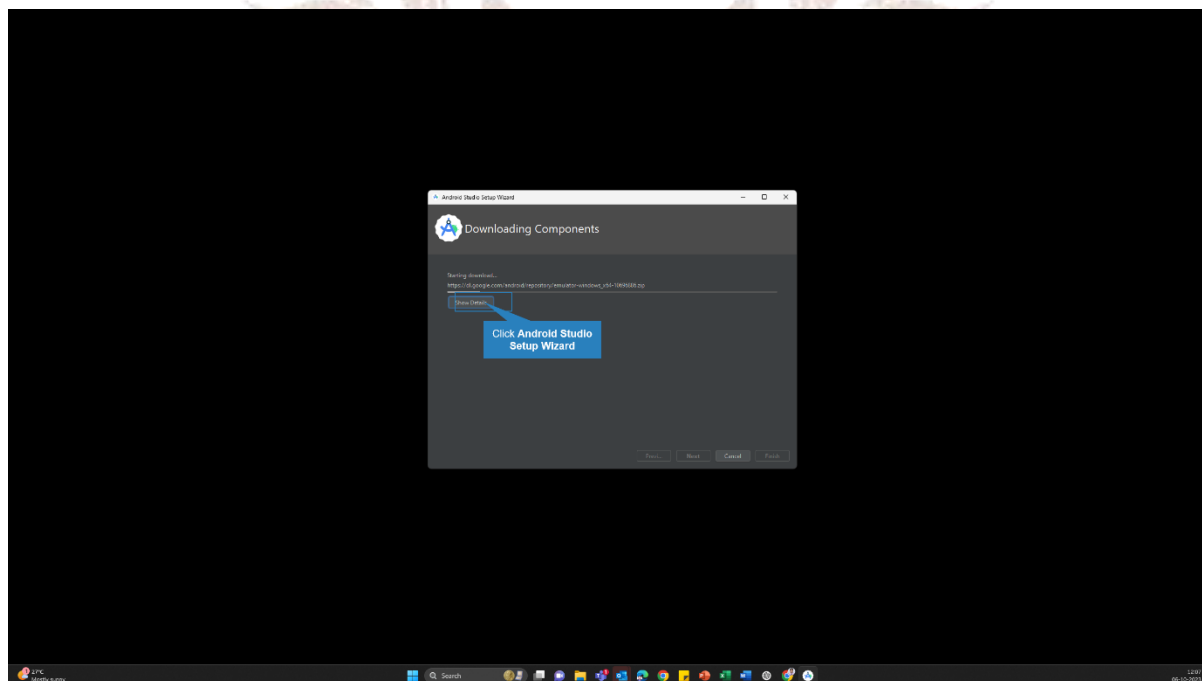
Step 15: Once the required licenses are selected click on Finish button.



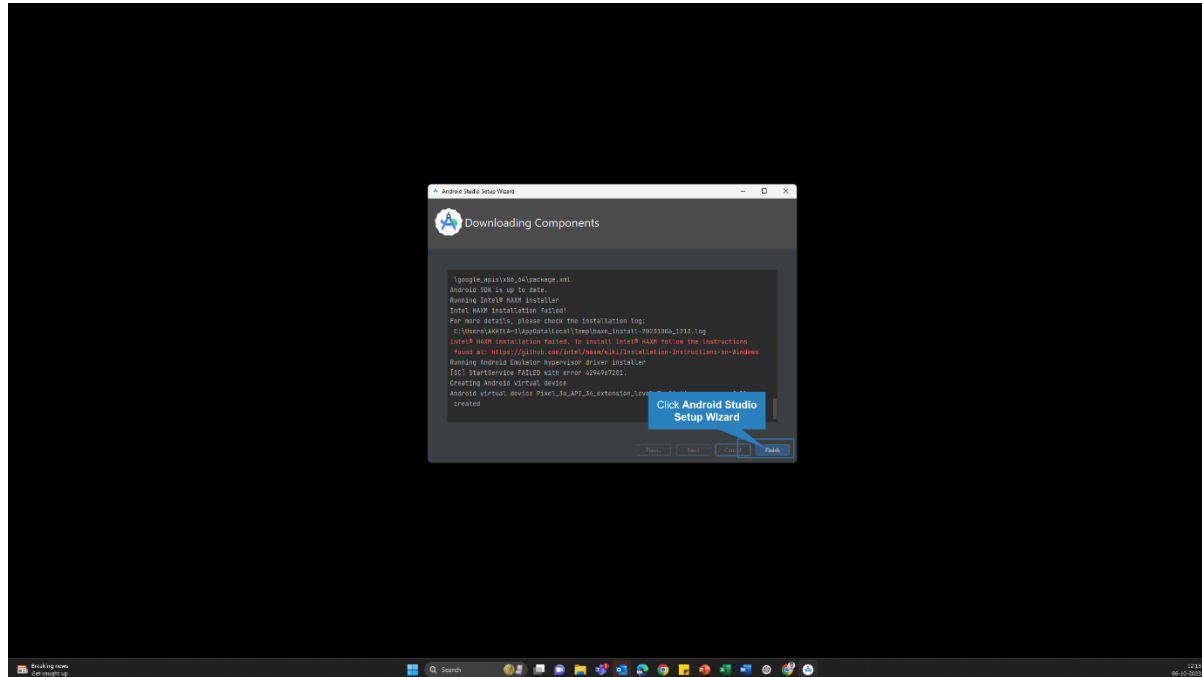
Step 16: After step 15, the required components start to get downloaded. Expand the Show details menu to see the details of components getting downloaded.



Step 17: Once all the required components are downloaded click on the finish button.



Step 18: Now, the environment to create projects on Android studio is ready and the developers can create the projects from scratch or to edit already created projects and version control system is also embedded onto the work station area.



## 6. SUMMARY

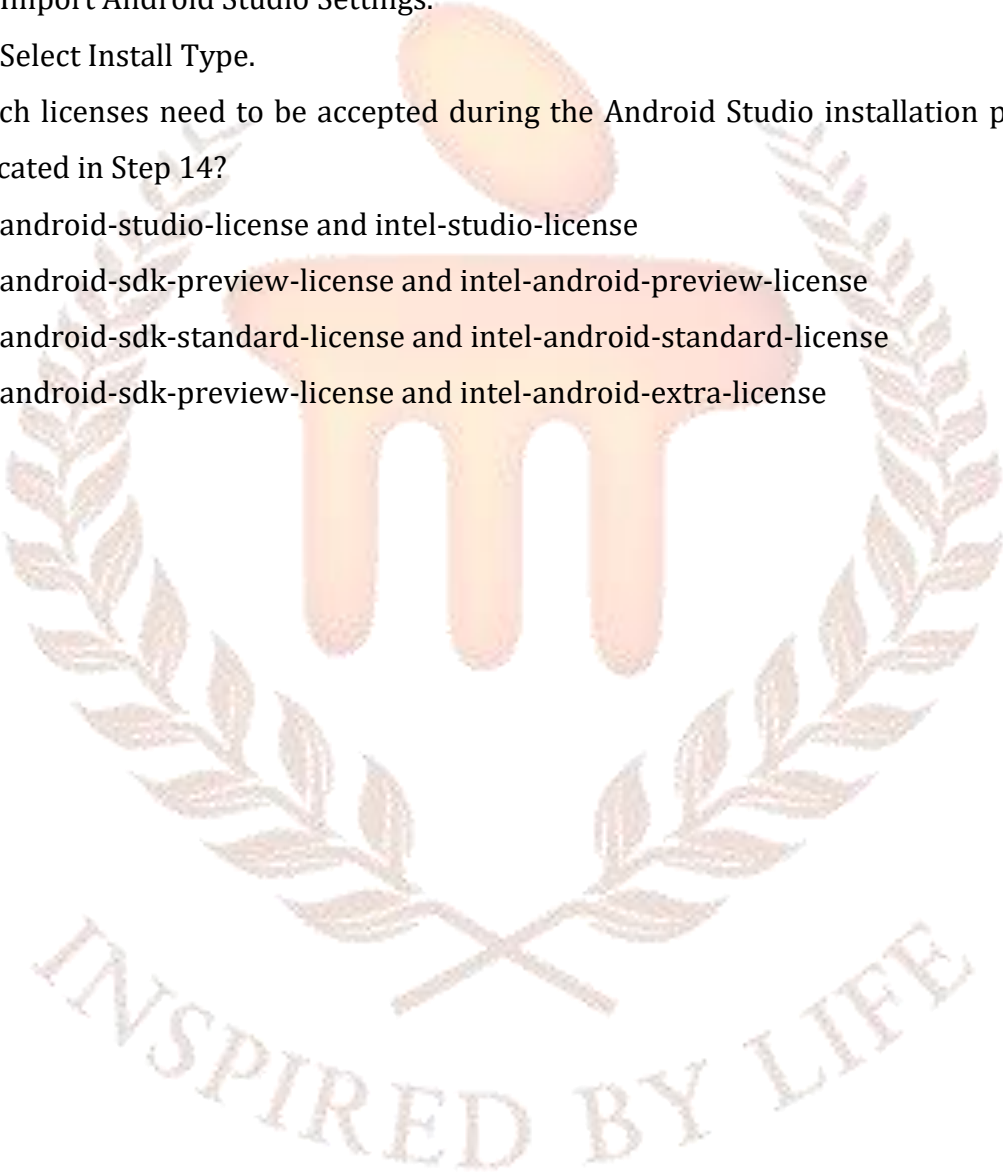
The unit tackles the fundamental aspects of Android Development, systematically guiding learners from system setup to advanced programming concepts in Java and Android Studio setup. Initially, it underscores the pivotal role system specifications play in Android development, delineating both the minimum and recommended system requirements, and providing a guide for emulator usage and additional tools. Subsequently, the unit dives into the Android SDK, elucidating its importance, installation, and advanced configuration, and followed by an in-depth exploration of Java. Herein, Java's basic concepts and syntax, including variables, data types, operators, control structures, and arrays, are meticulously unpacked. The unit culminates with a thorough walk-through of installing and configuring Android Studio, ensuring a well-rounded foundational understanding for aspiring Android developers.



## 7. SELF- ASSESSMENT QUESTIONS

1. What is the minimum recommended RAM for Android development?
  - a) 2GB
  - b) 4GB
  - c) 8GB
  - d) 6GB
2. Which of the following is NOT a recommended operating system for Android development?
  - a) Windows 10
  - b) macOS Catalina
  - c) LinuxWindows XP
3. What technology should be enabled in the BIOS for better emulator performance?
  - a) Hyper-Threading
  - b) Secure Boot
  - c) Intel VT-x or AMD-V
  - d) Power Management
4. What is the primary function of the Android SDK?
  - a) Web development
  - b) Game development
  - c) Android app development
  - d) iOS app development
5. Which tool in Android Studio manages SDK components?
  - a) AVD Manager
  - b) SDK Manager
  - c) Project Manager
  - d) Plugin Manager
6. What should you install for testing in emulators?
  - a) Only the latest SDK platform package
  - b) At least one system image
  - c) Android Web Tools

- d) Android Security Suite
7. Which of the following is a valid variable declaration in Java?
- a) `int 1age = 25;`
  - b) `float height; = 5.9`
  - c) `char initial = 'A';`
  - d) `boolean isActive == true;`
8. What does the `extends` keyword signify in Java?
- a) Encapsulation
  - b) Inheritance
  - c) Polymorphism
  - d) Initialization
9. What is used in Java to handle exceptions?
- a) if-else blocks
  - b) try-catch-finally blocks
  - c) for loops
  - d) switch statements
10. Which JDK version is recommended for Android development?
- a) JDK 5
  - b) JDK 6
  - c) JDK 7
  - d) JDK 8 or newer
11. What is Android Studio?
- a) A version control system
  - b) A type of Android device
  - c) The official IDE for Android development
  - d) A library for Java programming
12. What is the primary programming language for native Android app development?
- a) Python
  - b) C++
  - c) Java
  - d) JavaScript

13. During the installation process of Android Studio on Windows, what option are you prompted to select in Step 7?
- a) Choose UI theme.
  - b) Accept the License Agreement.
  - c) Import Android Studio Settings.
  - d) Select Install Type.
14. Which licenses need to be accepted during the Android Studio installation process as indicated in Step 14?
- a) android-studio-license and intel-studio-license
  - b) android-sdk-preview-license and intel-android-preview-license
  - c) android-sdk-standard-license and intel-android-standard-license
  - d) android-sdk-preview-license and intel-android-extra-license
- 

## 8. TERMINAL QUESTIONS

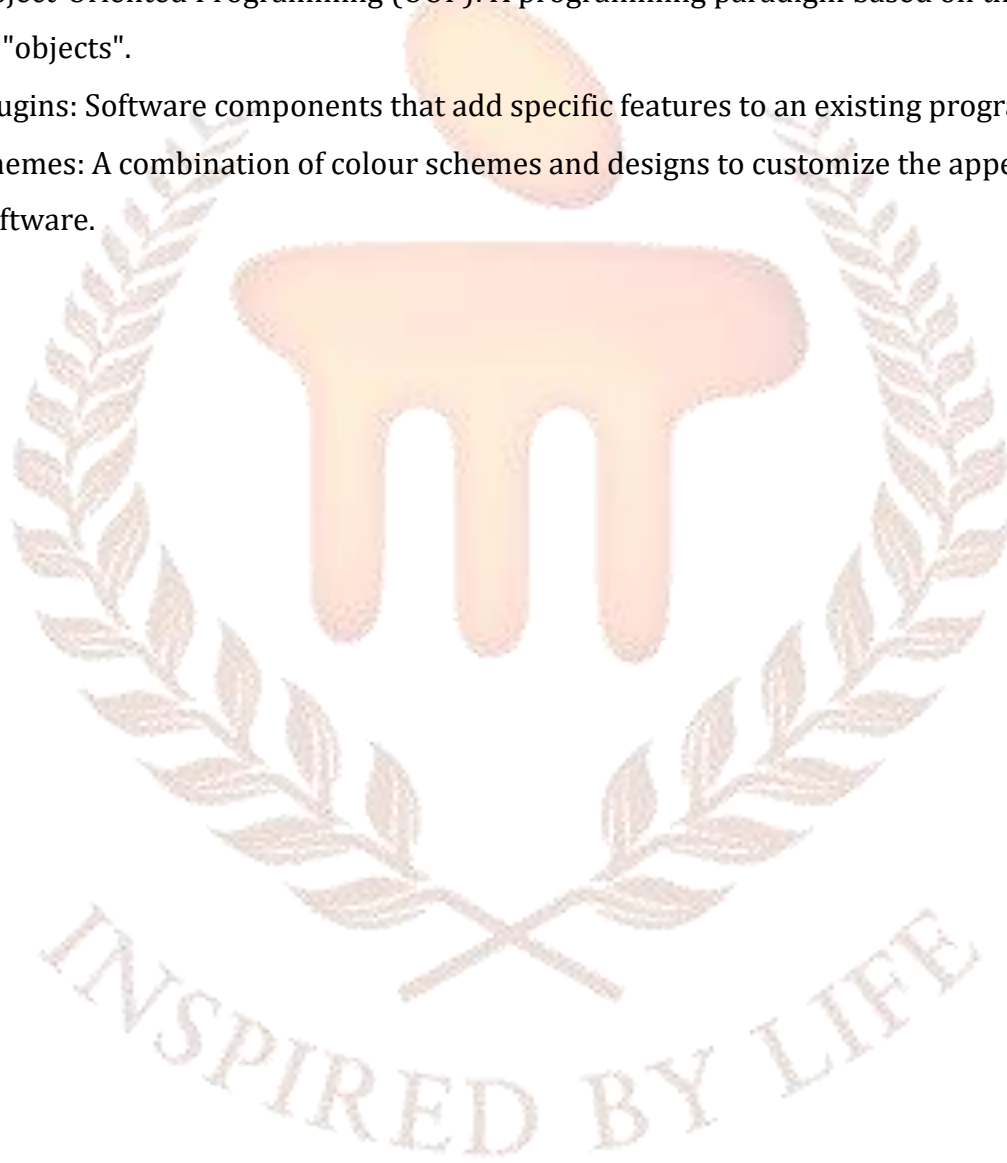
1. What is the significance of aligning your computer system with optimal specifications for Android development?
2. Why is it important to have a system tailored for Android development tasks?
3. What are the baseline system prerequisites for Android development?
4. What considerations should be made if a developer relies heavily on emulators for Android development?
5. What are the additional tools and software that can be included in the Android development toolkit?
6. Why might someone need to install the Android SDK separately even if they're using Android Studio?
7. What are the features and functionalities of the Android SDK Manager?
8. Why is Java considered pivotal to modern software development?
9. What are the characteristics that contribute to Java's popularity?
10. How can developers personalize the appearance of Android Studio?
11. What are the steps involved in the installation of Android Studio on a Windows system?

## 9. GLOSSARY

- **Android Software Development Kit (SDK):** A comprehensive set of development tools to create Android applications.
- **Emulator:** A software that offers a virtual representation of Android devices.
- **Java Development Kit (JDK):** A software development environment required for developing applications in Java.
- **Version Control Systems:** Tools, like Git, that help manage changes to source code over time.
- **Integrated Development Environment (IDE):** Software that consolidates the basic tools required for software testing and writing.
- **Android Studio:** The official IDE for Android development.
- **Hardware Virtualization:** A virtualization method at the hardware level.
- **Graphics Processing Unit (GPU):** A specialized electronic circuit designed to accelerate the image output in a frame buffer intended for display devices.
- **Java Virtual Machine (JVM):** A part of the Java Runtime Environment that compiles bytecode into machine language to be executed by the computer's processor.
- **Object-Oriented:** A programming paradigm based on the concept of "objects".
- **Multithreading:** The ability of a CPU, or a single core in a multi-core processor, to provide multiple threads of execution concurrently.
- **Primitive Data Types:** Basic data types provided by a programming language.
- **Reference Data Types:** Data types that store references to the memory location where data is kept.
- **Operators:** Symbols that perform operations on variables and values.
- **Control Structures:** Constructs that control the flow of program execution.
- **Array:** A data structure that can hold more than one value at a time.
- **Android Emulator:** A tool in the Android SDK that allows testing applications on virtual Android devices.
- **Android Virtual Device (AVD) Manager:** A tool to manage virtual devices for the Android Emulator.



- Proxy server: A proxy server is an intermediary server that sits between a client and the destination server.
- Proxy Settings: Configuration to allow software to communicate with the internet via a proxy server.
- Object-Oriented Programming (OOP): A programming paradigm based on the concept of "objects".
- Plugins: Software components that add specific features to an existing program.
- Themes: A combination of colour schemes and designs to customize the appearance of software.



## 10. ANSWERS

### 10.1 Self- Assessment Answers

1. b) 4GB
2. Windows XP
3. Intel VT-x or AMD-V
4. c) Android app development
5. b) SDK Manager
6. b) At least one system image
7. c) char initial = 'A';
8. b) Inheritance
9. b) try-catch-finally blocks
10. JDK 8 or newer
11. c) The official IDE for Android development
12. c) Java
13. c) Import Android Studio Settings.
14. android-sdk-preview-license and intel-android-extra-license.

### 10.2 Terminal Question Answers

1. Refer to section 2.1 - Introduction.
2. Refer to section 2.2 - Understanding the Importance of System Specifications.
3. Refer to section 2.4 - Baseline System Prerequisites.
4. Refer to section 2.6 - Special Considerations for Emulator Usage.
5. Refer to section 2.7 - Additional Development Tools and Software.
6. Refer to section 3.1 - Why Install the Android SDK?
7. Refer to section 3.5 - Using the Android SDK Manager.
8. Refer to section 4.1 - A Glimpse into Java's Genesis
9. Refer to section 4.2 - Why Java?
10. Refer to section 5.4 - Configuring Android Studio, under "Themes".
11. Refer to section 5.5 - Step-by-step installation of Android studio on Windows.

## 11. REFERENCES

- Android Developers Official Site. "System Requirements." Accessed 2023.
- Stack Overflow. "What are the system requirements for Android development?" Discussion Thread, 2020.
- Android Developers Official Site. "Install Android Studio." Accessed 2023.
- Oracle. "Java Documentation." Accessed 2023.
- Android Developers Official Site. "Develop apps in Java." Accessed 2023.

