Introduction to mobile application

UNIT 1

WHAT IS MOBILE APPLICATION DEVELOPMENT?

Mobile application development is the process of creating a software applications that run on mobile devices like smartphones and tablets .

WHAT IS WEB APPS?

A web app or web application is a software program that runs on a web server and is accessed through a web browser over the internet.

Example: Popular examples of web apps include email services like Gmail, social media platforms like facebook, online shopping websites like amazon and productivity tools like google docs.

Brief history of mobile technologies:

- **1)Early beginnings :**
- the journey of mobile technology began in **the early 20th century** with the **invention of radio communication**
- Development of mobile **radio telephones in the 1940s**
- It was primarily used in the **vehicles with the limited coverage and functionality.**

- **2)First generation (1G):**
- **invented in the 1980s** marked as the **beginning of mobile telephony**.
- **1G networks were analog** and allowed for **voice communication** through **mobile phones.**
- The launch of the **Motorola DynaTAC 8000X in 1983**
- The first commercially available **mobile phone**, was a **landmark event despite its large size** and **limited battery life .**

Brief history of mobile technologies:

- **3) Second generation (2G) :**
- **1990s** it was invented.
- Which were **digital and offered significant improvements in voice quality and capacity.**
- 2G networks introduced the **global system for mobile communication (GSM)** and it became the **std for mobile communication worldwide**.
- Introduction of **SMS came into existing** .

- **4) Third generation (3G):**
- in **early 2000**
- Networks **enable faster data transmission** and **improved internet access on mobile devices**.
- Supported **multimedia messaging, video calls, and mobile internet browsing**
- It showed the **way for the development of more advance applications**

Brief history of mobile technologies:

- **5) Fourth Generation (4G) and the rise of smartphones:**
- Late 2000s
- Revolution of mobile technology with **high speed internet access** to enable seamless **streaming of high-definition video , faster downloads** and enable the **mobile gaming experiences.**
- 4G offering high data speeds and lower latency.
- In this period the **smart phones**, integrating powerful processors, high resolution displays and wide range of sensors were introduced

6) Fifth generation (5G) and beyond) :

- 5g represents the **latest advancement in mobile technology**
- Promising **ultra high fast speeds**
- Low latency
- Massive connectivity to support emerging technologies like **internet of things (IOT), AR and autonomous vehicles**.
- 5g is set to revolutionize industries by enabling **smart cities, autonomous vehicles and advance health care solution .**

DIFFERENT MOBILE TECHNOLOGIES :

- 1) cellular networks:
- Cellular networks are the foundation of modern mobile communation
- It enables us to make use of phone calls , send text message and access the internet from our mobile devices.
- Generations to meet the increasing demands for better performance, connectivity .
- 1G
- 2G
- 3G
- 4G
- 5G

MOBILE OPERATING SYSTEMS (OS)

- OS are the software platforms that manage the operations of mobile devices such as smartphones, tablets and smartphones by providing the essential software frameworks that helps the devices to run and function smoothly .
- **Some key mobile operating systems:**
- **ANDROID :**
- developed by google
- It is widely used mobile OS globally.
- It is based on linux kernel and its an open source
- Allowing manufactures to customize it for there devices.
- Available in google play store.

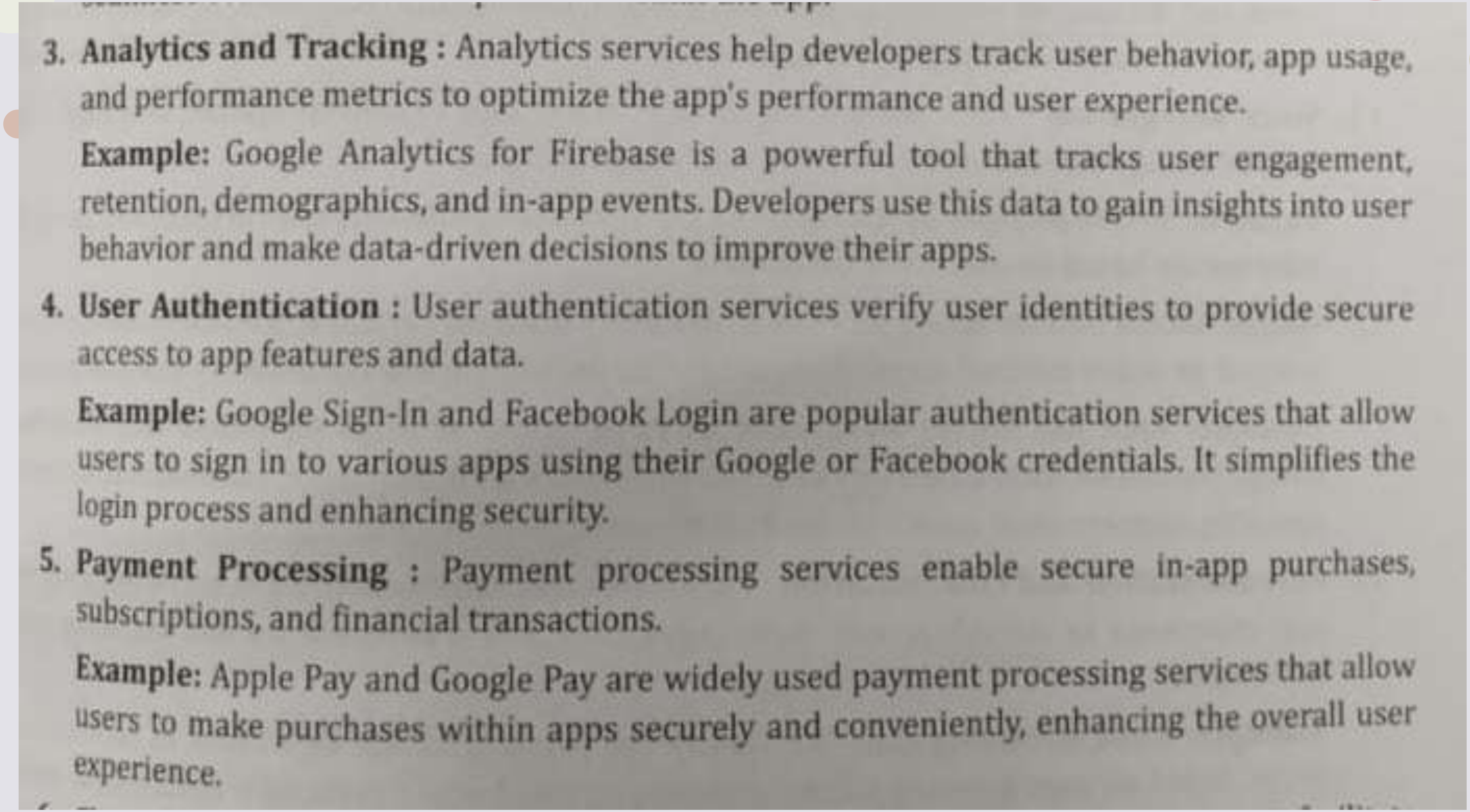
2) IOS:

- Developed by **apple**
- Ios is an OS which is exclusively for apple users
- It includes iphone, ipad, ipod touch.
- IOS is known as for its smooth performance ,security features and tight integration with apple's hardware and software ecosystem
- Apps can be installed only through apple store only .
- 1) windows mobile
- 2) harmony OS
- 3) blackberry OS

KEY MOBILE APPLICATION SERIVES:

- **1) PUSH NOTIFICATIONS** : push information are essential for engaging users and delivering timely information.
- They allow apps to send messages to users even when the app is not actively in use.
- **Eg**: facebook, whatsapp, Instagram , etc

- **2)IN-APP MESSAGING** : in app messaging enables users to communicate within the app to enhance the interaction among the users.
- **Eg** : all messageing apps – whats app,, facebook, …services messaages are sending msg , voice , video , image



3. **Analytics and Tracking :** Analytics services help developers track user behavior, app usage, and performance metrics to optimize the app's performance and user experience.

   Example: Google Analytics for Firebase is a powerful tool that tracks user engagement, retention, demographics, and in-app events. Developers use this data to gain insights into user behavior and make data-driven decisions to improve their apps.

4. **User Authentication :** User authentication services verify user identities to provide secure access to app features and data.

   Example: Google Sign-In and Facebook Login are popular authentication services that allow users to sign in to various apps using their Google or Facebook credentials. It simplifies the login process and enhancing security.

5. **Payment Processing :** Payment processing services enable secure in-app purchases, subscriptions, and financial transactions.

   Example: Apple Pay and Google Pay are widely used payment processing services that allow users to make purchases within apps securely and conveniently, enhancing the overall user experience.

6. **Cloud Storage :** Cloud storage services enable apps to store data on remote servers to facilitate data synchronization and backup across multiple devices.

Example: Google Drive and iCloud offer cloud storage solutions that allow users to store and sync documents, photos, and other files across their devices to ensure data accessibility and backup.

7. **Location Services :** Location services empower apps to provide location-based functionalities such as navigation, geotagging, and location-based notifications.

Example: Apps like Google Maps and Uber utilize location services to offer navigation, real-time location tracking, and personalized services based on the user's location to enhance user experience and convenience.

8. **Social Media Integration :** Social media integration services enable apps to connect with social platforms, allowing users to share content and interact with their social networks seamlessly.

Example: Social media apps like Instagram and TikTok integrate with platforms like Facebook and Twitter, enabling users to share photos, videos, and updates directly from the app to their social media accounts, enhancing user engagement and reach.

9. **Cloud Messaging :** Cloud messaging services facilitate sending messages and notifications to users across different devices and platforms.

   Example: Firebase Cloud Messaging (FCM) is a robust cloud messaging service that enables developers to send notifications and messages to users on iOS, Android, and web applications, ensuring effective communication and engagement.

10. **Data Synchronization :** Data synchronization services ensure that user data remains consistent and up-to-date across multiple devices and platforms.

    Example: Apps like Evernote and Microsoft OneNote utilize data synchronization to keep notes and documents updated across smartphones, tablets, and computers, allowing users to access their data seamlessly across devices.

11. **Voice Recognition :** Voice recognition services allow apps to convert spoken language into text, enabling voice commands and voice-activated features.

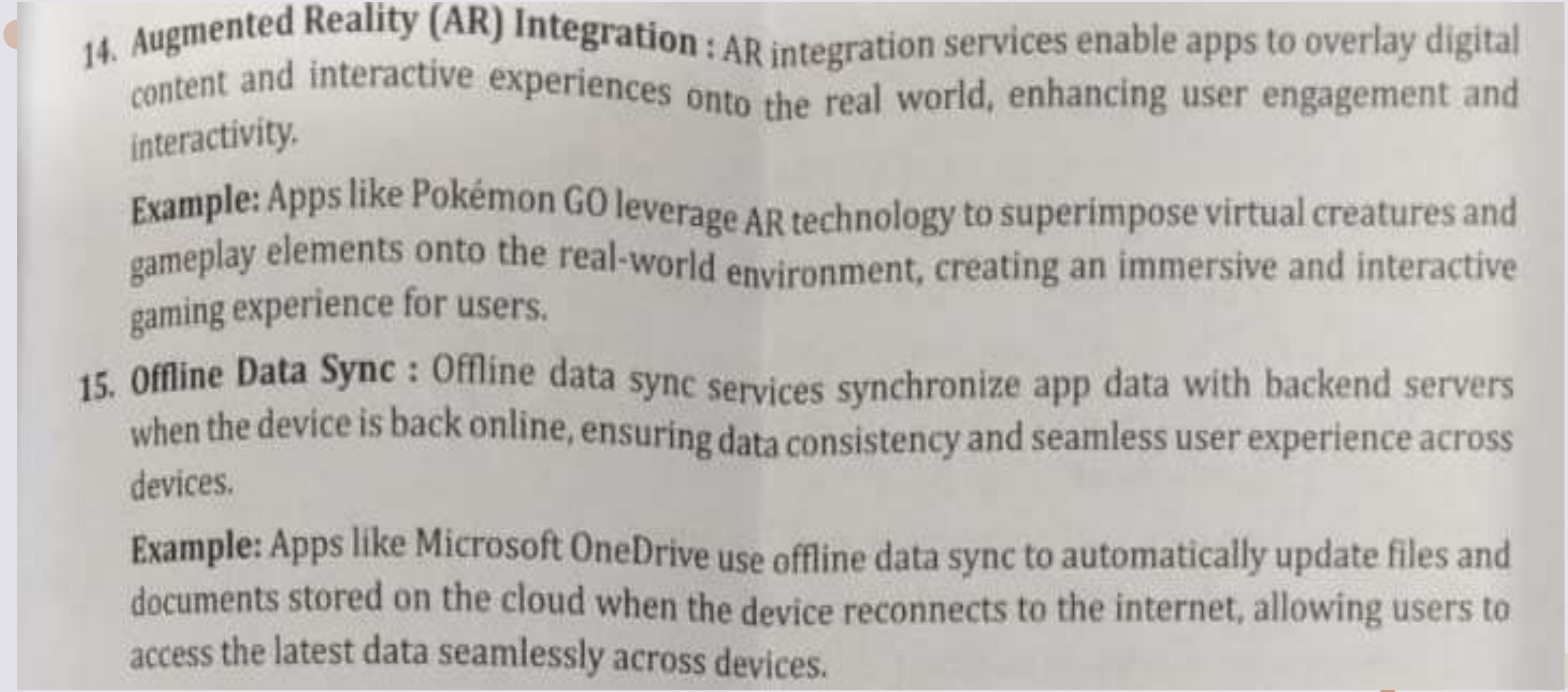
    Example: Siri (Apple) and Google Assistant use voice recognition to perform tasks and provide information based on user voice commands.

12. **Offline Access :** Offline access services enable users to access certain app features and content without an active internet connection, enhancing usability in low-connectivity environments.

Example: Apps like Google Maps allow users to download maps for offline use, enabling navigation and location-based services even when users are offline or have limited connectivity, ensuring uninterrupted access to essential features.

13. **Personalization and Customization :** Personalization and customization services tailor the app experience to individual user preferences, providing a personalized and engaging user experience.

Example: Music streaming apps like Spotify use personalization algorithms to recommend music based on user listening habits, preferences, and history, creating a customized music experience for each user and enhancing user engagement.

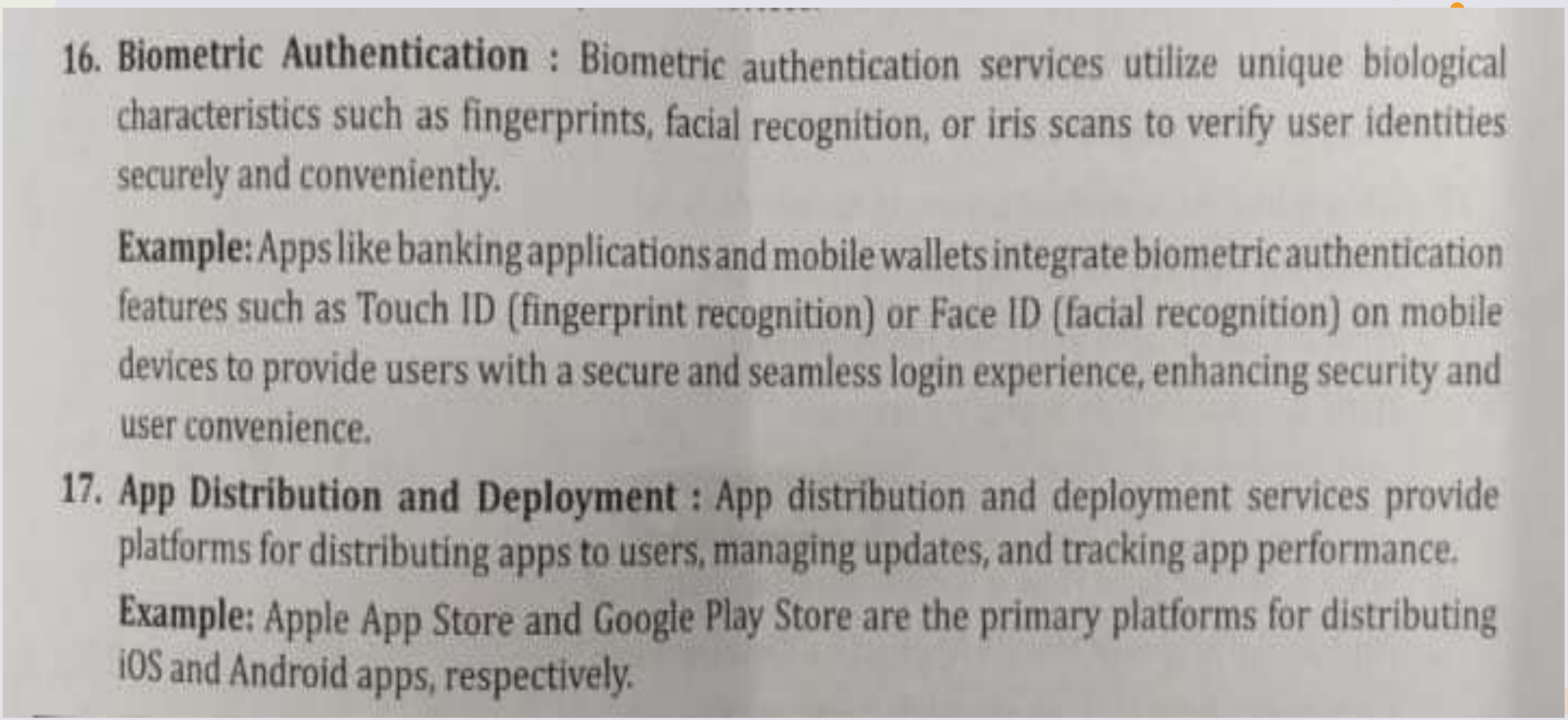


14. **Augmented Reality (AR) Integration :** AR integration services enable apps to overlay digital content and interactive experiences onto the real world, enhancing user engagement and interactivity.

    Example: Apps like Pokémon GO leverage AR technology to superimpose virtual creatures and gameplay elements onto the real-world environment, creating an immersive and interactive gaming experience for users.

15. **Offline Data Sync :** Offline data sync services synchronize app data with backend servers when the device is back online, ensuring data consistency and seamless user experience across devices.

    Example: Apps like Microsoft OneDrive use offline data sync to automatically update files and documents stored on the cloud when the device reconnects to the internet, allowing users to access the latest data seamlessly across devices.



16. **Biometric Authentication :** Biometric authentication services utilize unique biological characteristics such as fingerprints, facial recognition, or iris scans to verify user identities securely and conveniently.

    Example: Apps like banking applications and mobile wallets integrate biometric authentication features such as Touch ID (fingerprint recognition) or Face ID (facial recognition) on mobile devices to provide users with a secure and seamless login experience, enhancing security and user convenience.

17. **App Distribution and Deployment :** App distribution and deployment services provide platforms for distributing apps to users, managing updates, and tracking app performance.

    Example: Apple App Store and Google Play Store are the primary platforms for distributing iOS and Android apps, respectively.

THE ANDROID APPLICATION COMPONENTS :

- Android components can be categorized into PRIMARY COMPONENTS (core components) and SECONDARY COMPOUNDS (Non-core) based on there fundamental roles and supplementary functionalities within an application.

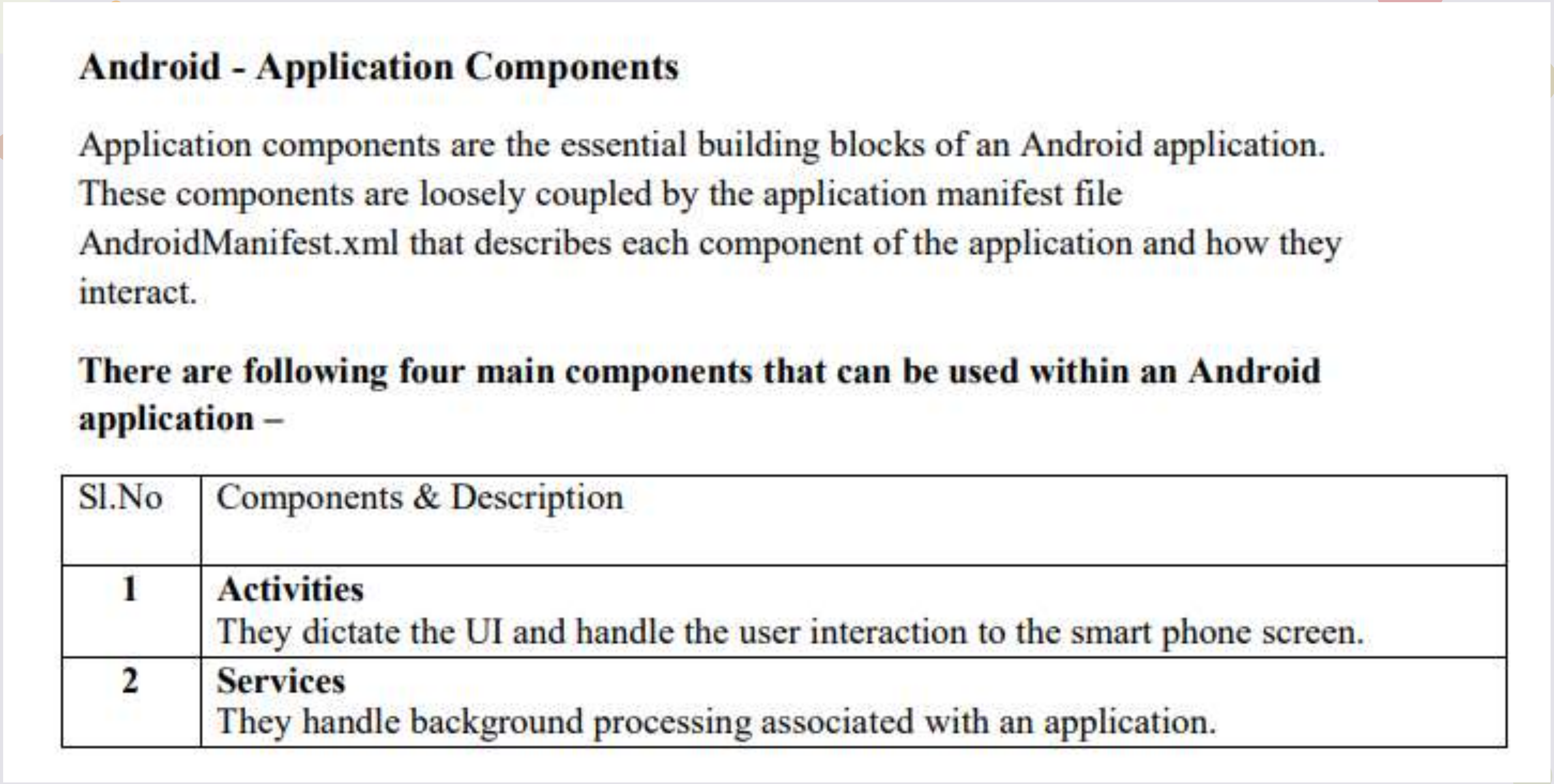
- ***Primary (or) core components :***

✓Core component are crucial for the basic functionality and structure of an android application

✓They handle the main processes, user interactions and essential background tasks that make the app operational.

Primary or core components :

- **1) Activities** :
- *activities represents the user interface (UI) of an application.*
- *They are responsible for the users and handling user interactions.*
- **2) services :**
- 3)Broadcast receiver
- 4) content Providers : CRUD –create , read, update, delete



Android - Application Components

Application components are the essential building blocks of an Android application. These components are loosely coupled by the application manifest file AndroidManifest.xml that describes each component of the application and how they interact.

There are following four main components that can be used within an Android application –

Sl.No	Components & Description
1	**Activities** They dictate the UI and handle the user interaction to the smart phone screen.
2	**Services** They handle background processing associated with an application.

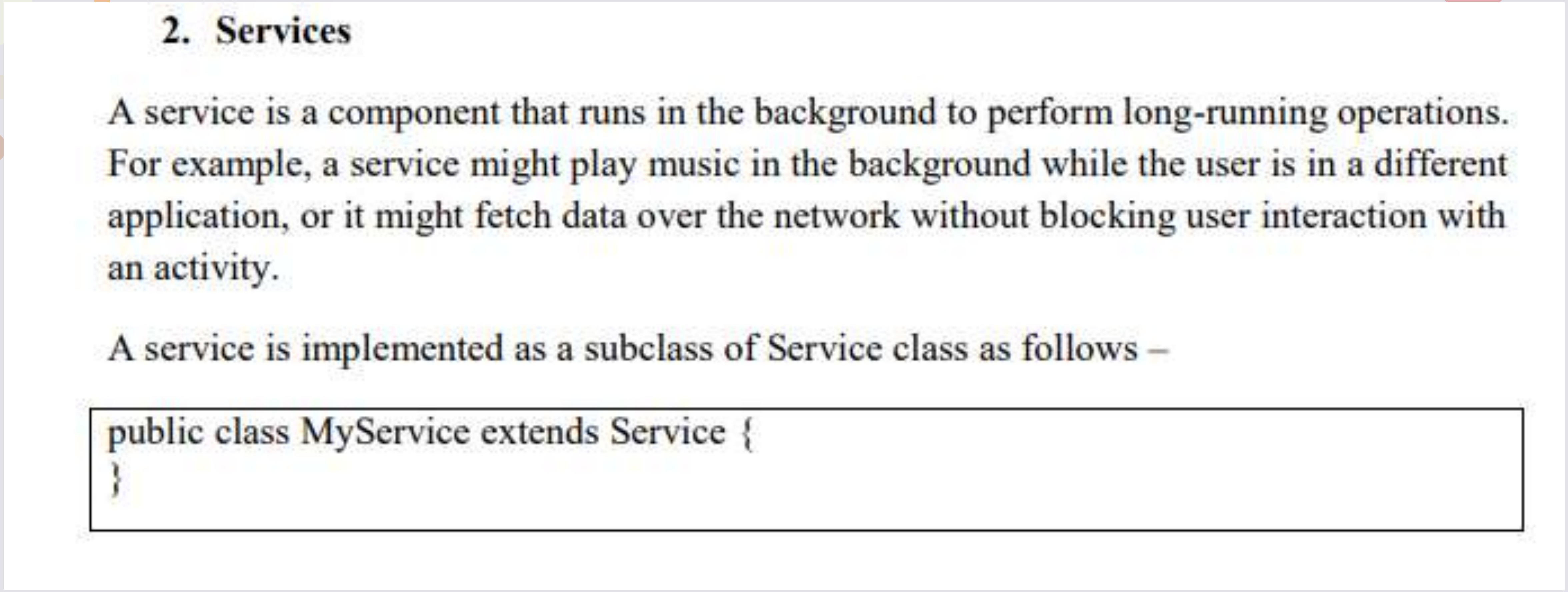
3	**Broadcast Receivers** They handle communication between Android OS and applications.
4	**Content Providers** They handle data and database management issues.

1. Activities

An activity represents a single screen with a user interface, in-short Activity performs actions on the screen. For example, an email application might have one activity that shows a list of new emails, another activity to compose an email, and another activity for reading emails. If an application has more than one activity, then one of them should be marked as the activity that is presented when the application is launched.

An activity is implemented as a subclass of Activity class as follows –

```
public class MainActivity extends Activity {
}
```



2. Services

A service is a component that runs in the background to perform long-running operations. For example, a service might play music in the background while the user is in a different application, or it might fetch data over the network without blocking user interaction with an activity.

A service is implemented as a subclass of Service class as follows –

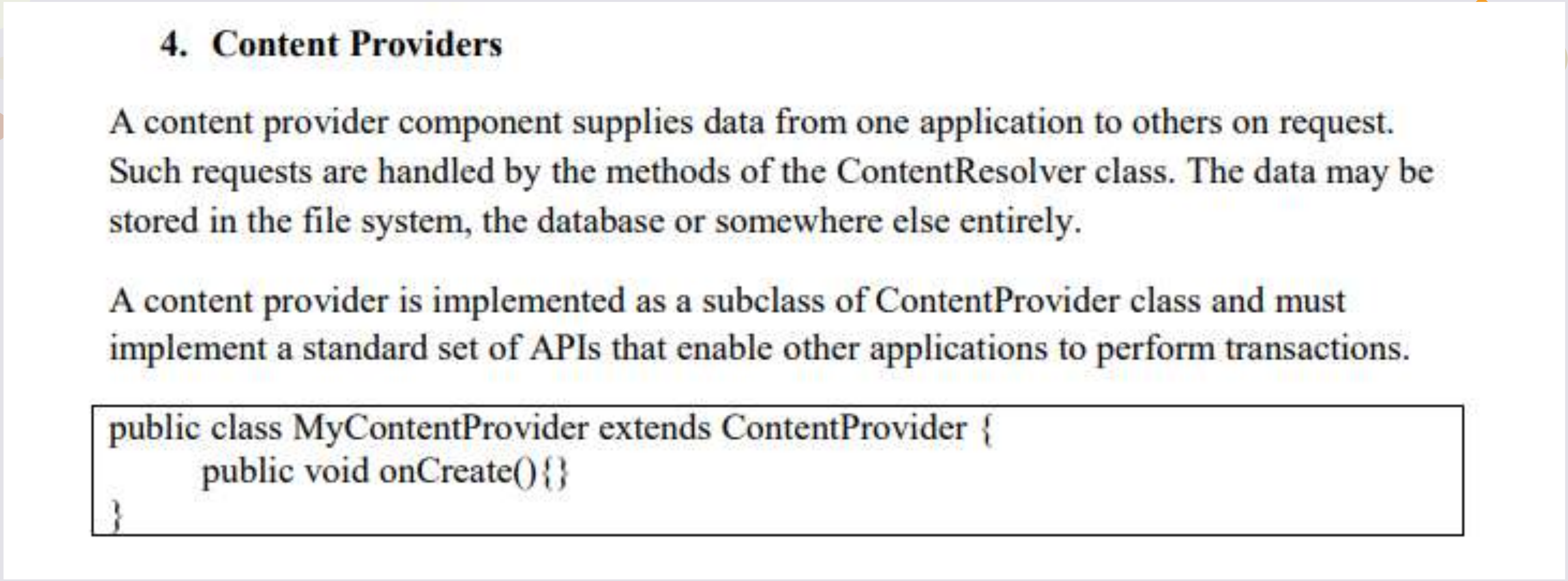
```
public class MyService extends Service {
}
```

3. Broadcast Receivers

Broadcast Receivers simply respond to broadcast messages from other applications or from the system. For example, applications can also initiate broadcasts to let other applications know that some data has been downloaded to the device and is available for them to use, so this is broadcast receiver who will intercept this communication and will initiate appropriate action.

A broadcast receiver is implemented as a subclass of BroadcastReceiver class and each message is broadcaster as an Intent object.

```
public class MyReceiver extends BroadcastReceiver {
        public void onReceive (context, intent){}
}
```

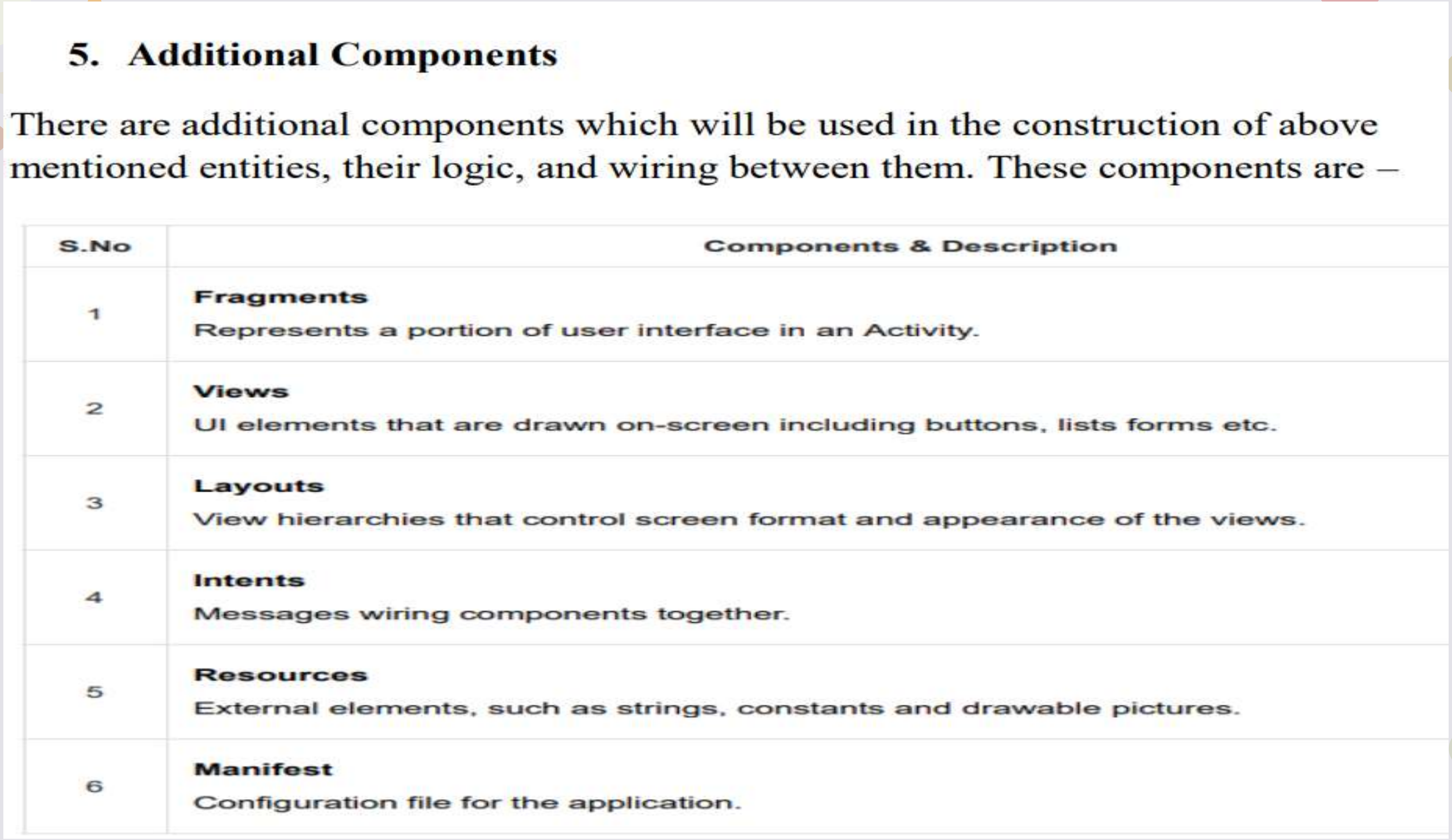


4. Content Providers

A content provider component supplies data from one application to others on request. Such requests are handled by the methods of the ContentResolver class. The data may be stored in the file system, the database or somewhere else entirely.

A content provider is implemented as a subclass of ContentProvider class and must implement a standard set of APIs that enable other applications to perform transactions.

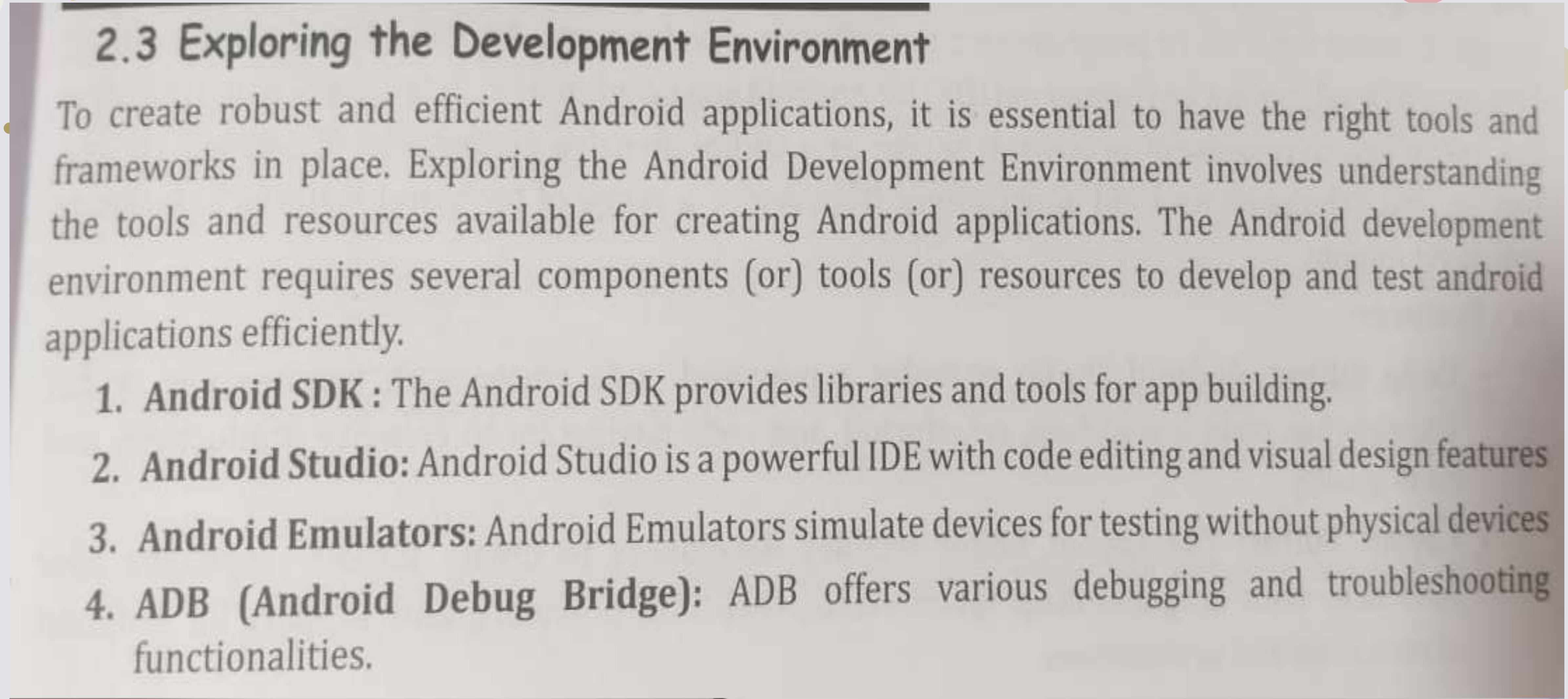
```
public class MyContentProvider extends ContentProvider {
        public void onCreate(){}
}
```



5. Additional Components

There are additional components which will be used in the construction of above mentioned entities, their logic, and wiring between them. These components are –

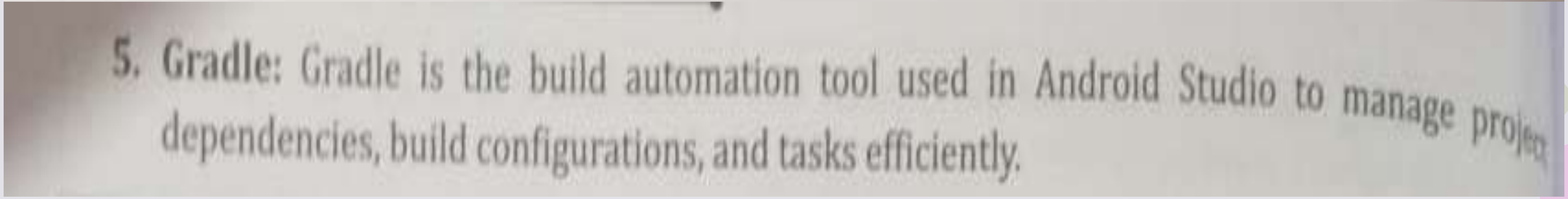
S.No	Components & Description
1	**Fragments** Represents a portion of user interface in an Activity.
2	**Views** UI elements that are drawn on-screen including buttons, lists forms etc.
3	**Layouts** View hierarchies that control screen format and appearance of the views.
4	**Intents** Messages wiring components together.
5	**Resources** External elements, such as strings, constants and drawable pictures.
6	**Manifest** Configuration file for the application.



2.3 Exploring the Development Environment

To create robust and efficient Android applications, it is essential to have the right tools and frameworks in place. Exploring the Android Development Environment involves understanding the tools and resources available for creating Android applications. The Android development environment requires several components (or) tools (or) resources to develop and test android applications efficiently.

1. **Android SDK :** The Android SDK provides libraries and tools for app building.
2. **Android Studio:** Android Studio is a powerful IDE with code editing and visual design features
3. **Android Emulators:** Android Emulators simulate devices for testing without physical devices
4. **ADB (Android Debug Bridge):** ADB offers various debugging and troubleshooting functionalities.



5. **Gradle:** Gradle is the build automation tool used in Android Studio to manage proje dependencies, build configurations, and tasks efficiently.

2.3.1 Android SDK

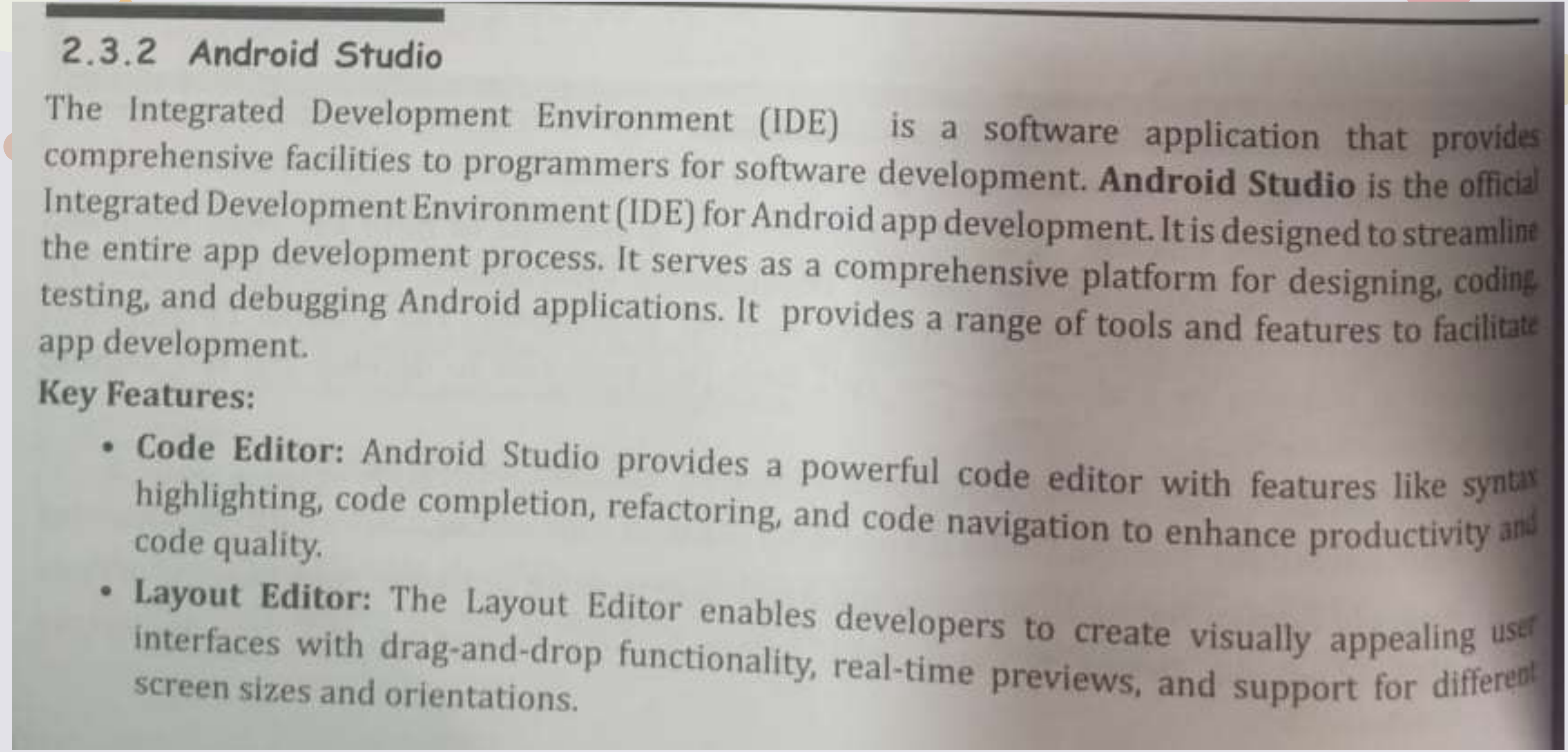
The Android SDK is a set of tools, libraries, and resources provided by Google to develop Android applications. It serves as the foundation for Android app development and provides essential components and APIs for building feature-rich apps.

What is an Android SDK?

An Android SDK (Software Development Kit) is a set of tools, libraries, and documentation provided by Google to help developers create applications for the Android platform. It includes everything developers need to start building Android apps, such as APIs for interacting with the device hardware, emulator for testing apps, and various other tools for debugging and profiling.

Key Features:

- **Libraries and APIs:** Provides access to a wide range of libraries and APIs for implementing various functionalities in Android apps. These libraries cover various aspects of Android development, from UI components to data storage and networking.
- **Development Tools:** Includes compilers, debuggers, and build tools for writing, compiling and debugging Android applications.
- **Documentation:** Offers detailed documentation on APIs, best practices, and development guidelines for Android app development.
- **Sample Code and Templates:** Provides sample code snippets and templates to assist developers in implementing common tasks and features.
- **Updates and Support:** Regularly updated by Google to incorporate new features, enhancements, and compatibility with the latest Android versions.



2.3.2 Android Studio

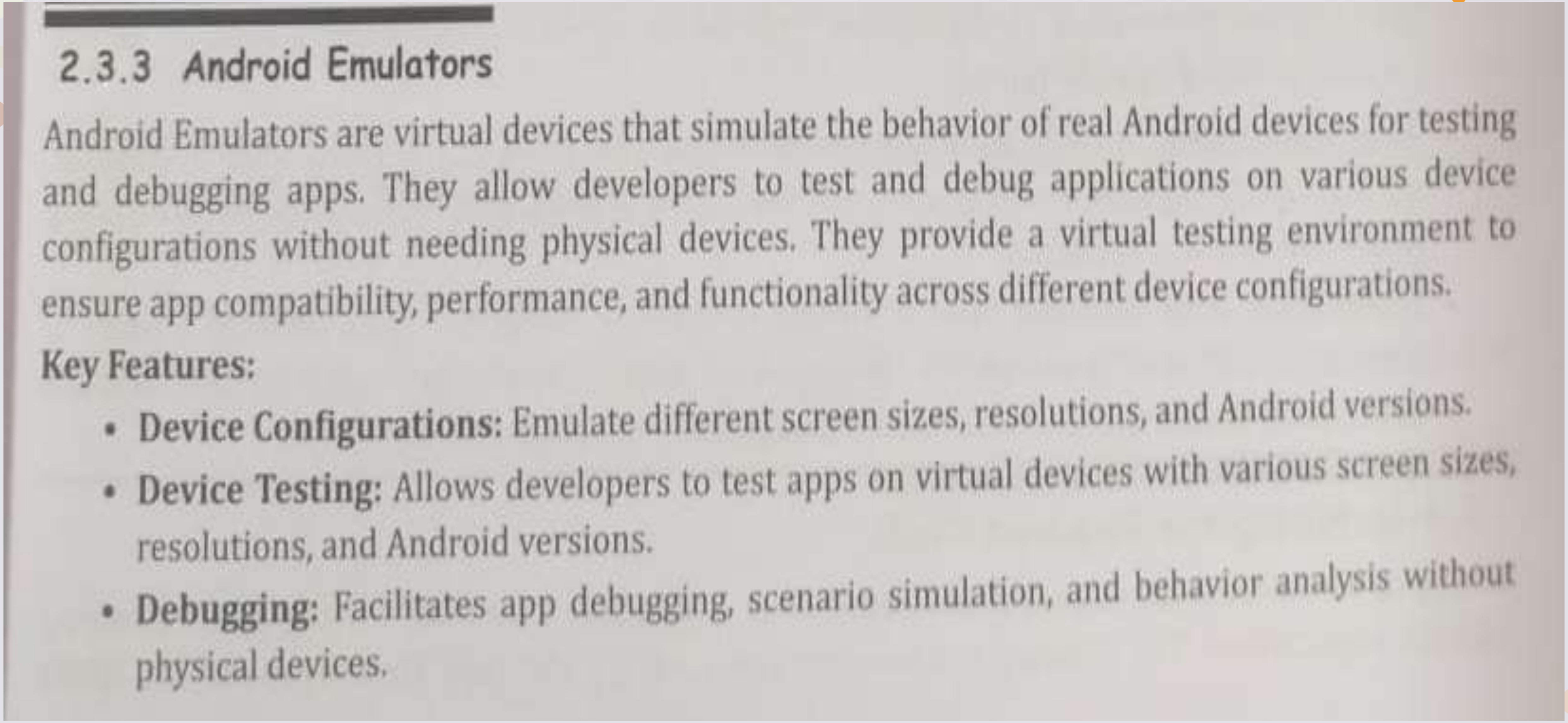
The Integrated Development Environment (IDE) is a software application that provides comprehensive facilities to programmers for software development. **Android Studio** is the official Integrated Development Environment (IDE) for Android app development. It is designed to streamline the entire app development process. It serves as a comprehensive platform for designing, coding, testing, and debugging Android applications. It provides a range of tools and features to facilitate app development.

Key Features:

- **Code Editor:** Android Studio provides a powerful code editor with features like syntax highlighting, code completion, refactoring, and code navigation to enhance productivity and code quality.
- **Layout Editor:** The Layout Editor enables developers to create visually appealing user interfaces with drag-and-drop functionality, real-time previews, and support for different screen sizes and orientations.

- **SDK Manager :** The Android SDK Manager is a tool within Android Studio that allows developers to download, install, and manage different versions of the Android SDK, as well as other essential tools and components.
- **Device Manager :** The Device Manager is a tool within Android Studio that allows developers to create and manage virtual devices that simulate physical Android devices. These virtual devices are used to test and debug applications on various configurations and Android versions without needing physical devices.
- **APK Analyzer:** Helps analyze APK size, contents, and dependencies to optimize app performance and reduce file size.
- **Built-in Emulator:** The built-in Android Emulator allows developers to test their apps on virtual devices with various configurations, screen sizes, and Android versions for comprehensive testing.
- **Device Testing:** Developers can test their apps on physical devices connected via USB for real-world testing scenarios, in addition to the Android Emulator, to ensure app compatibility and performance.

- **Version Control Integration:** Android Studio supports version control systems like Git, enabling developers to manage source code, track changes, and collaborate with team members efficiently within the IDE.
- **Extensive Plugin Ecosystem:** Android Studio offers a wide range of plugins and extensions from the Plugin Marketplace to extend functionality, integrate with external tools, and customize the development environment according to specific requirements.
- **Performance Profiling Tools:** Android Studio offers performance profiling tools to analyze app performance, identify bottlenecks, and optimize CPU, memory, and network usage for better user experience.
- **Build Tools:** Android Studio integrates the Gradle build system for automating build tasks, managing dependencies, and configuring build variants to streamline the app development proces
- **Integrated Debugger:** An integrated debugger allows developers to identify and fix issues in their code by setting breakpoints, inspecting variables, and stepping through code execution.

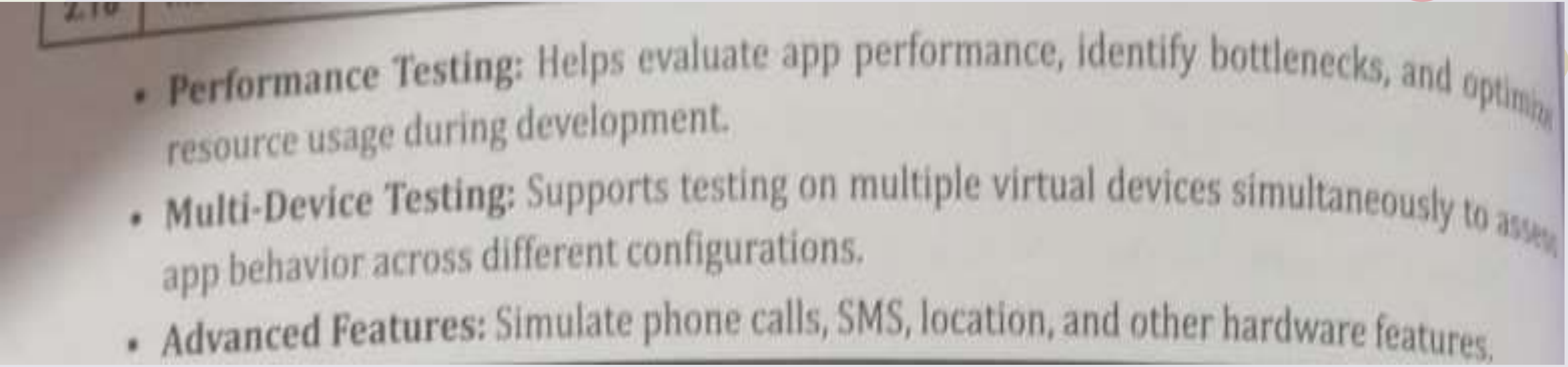


2.3.3 Android Emulators

Android Emulators are virtual devices that simulate the behavior of real Android devices for testing and debugging apps. They allow developers to test and debug applications on various device configurations without needing physical devices. They provide a virtual testing environment to ensure app compatibility, performance, and functionality across different device configurations.

Key Features:

- **Device Configurations:** Emulate different screen sizes, resolutions, and Android versions.
- **Device Testing:** Allows developers to test apps on virtual devices with various screen sizes, resolutions, and Android versions.
- **Debugging:** Facilitates app debugging, scenario simulation, and behavior analysis without physical devices.



2.10

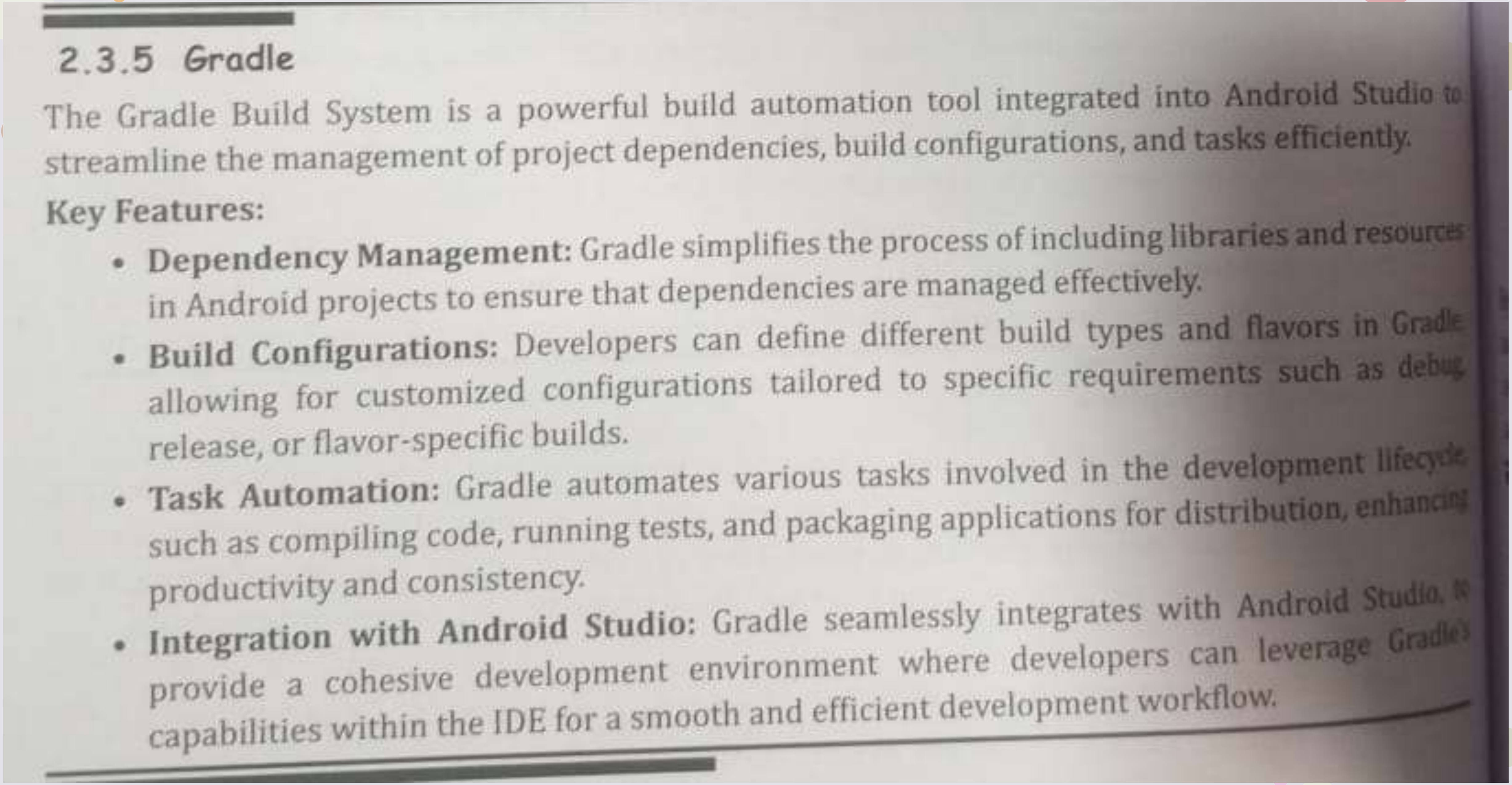
- **Performance Testing:** Helps evaluate app performance, identify bottlenecks, and optimi resource usage during development.
- **Multi-Device Testing:** Supports testing on multiple virtual devices simultaneously to asses app behavior across different configurations.
- **Advanced Features:** Simulate phone calls, SMS, location, and other hardware features.

2.3.4 ADB (Android Debug Bridge)

The Android Debug Bridge (ADB) is a versatile command-line tool that serves as a bridge between a development machine and an Android device or emulator. It offers a range of debugging and troubleshooting functionalities to help in the development and testing of Android applications.

Key Functions:

- **Installing and Debugging Apps:** ADB allows developers to install and debug applications directly on connected Android devices for efficient testing and troubleshooting.
- **Accessing Device's Shell:** It provides access to the device's shell, allowing developers to perform advanced troubleshooting tasks and execute commands directly on the device.
- **Transferring Files:** ADB facilitates the seamless transfer of files between a computer and an Android device to simplify the tasks such as moving resources, logs, or APK files for testing purposes.
- **Managing Device State and Permissions:** Developers can use ADB to manage the state of the device, change permissions, and simulate various scenarios for testing different aspects of their applications.



2.3.5 Gradle

The Gradle Build System is a powerful build automation tool integrated into Android Studio to streamline the management of project dependencies, build configurations, and tasks efficiently.

Key Features:

- **Dependency Management:** Gradle simplifies the process of including libraries and resources in Android projects to ensure that dependencies are managed effectively.
- **Build Configurations:** Developers can define different build types and flavors in Gradle allowing for customized configurations tailored to specific requirements such as debug, release, or flavor-specific builds.
- **Task Automation:** Gradle automates various tasks involved in the development lifecycle, such as compiling code, running tests, and packaging applications for distribution, enhancing productivity and consistency.
- **Integration with Android Studio:** Gradle seamlessly integrates with Android Studio, to provide a cohesive development environment where developers can leverage Gradle's capabilities within the IDE for a smooth and efficient development workflow.