RESEARCH

1. a. What is Android? Who created it? What are Android Apps?

Android is a mobile operating system based on a modified version of the Linux kernel and other open source software, designed primarily for touchscreen mobile devices such as smartphones and tablets. The hardware that supports android software is based on ARM architecture platform. Android is developed by a consortium of developers known as the Open Handset Alliance, with the main contributor and commercial marketer being Google. The android is an open source operating system means that it's free and anyone can use it.
Android Inc. was founded in Palo Alto, California, in October 2003 by Andy Rubin, Rich Miner, Nick Sears, and Chris White. Rubin described the Android project as "tremendous potential in developing smarter mobile devices that are more aware of its owner's location and preferences".
In July 2005, Google acquired Android Inc. for at least $50 million. Its key employees, including Rubin, Miner and White, joined Google as part of the acquisition. At Google, the team led by Rubin developed a mobile device platform powered by the Linux kernel.
The first commercially available smartphone running Android was the HTC Dream, also known as T-Mobile G1, announced on September 23, 2008.
An Android app is a software application running on the Android platform. Because the Android platform is built for mobile devices, a typical Android app is designed for a smartphone or a tablet PC running on the Android OS.

1. b. What is the software used in the development of Android Apps?

Android apps can be written using Kotlin, Java, and C++ languages using the Android software development kit (SDK), while using other languages is also possible.
In December 2014, Google released Android Studio, based on IntelliJ IDEA, as its primary IDE for Android application development.

1. c. Which are the languages commonly used in the development of android apps? Which language does InstiApp use?

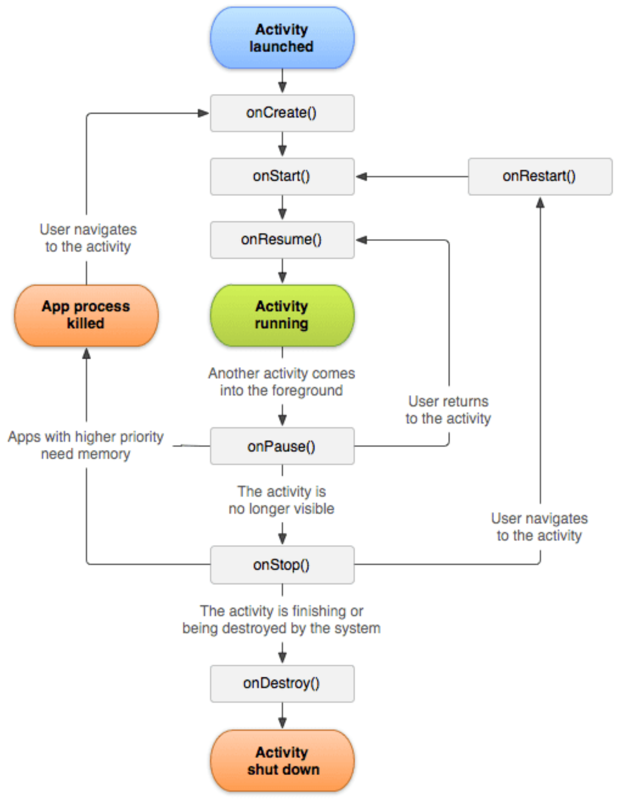
Java is the official language for Android App Development and consequently, it is the most used language as well.
Kotlin is a cross-platform programming language that has also been introduced as a secondary "official" Java language in 2017.
C++ can be used for Android App Development using the Android Native Development Kit(NDK).
C# is quite similar to Java and so it is ideal for Android App Development.
Android apps can be created using HTML, CSS, and JavaScript using the Adobe PhoneGap framework that is powered by Apache Cordova.
Language used for InstiApp: InstiApp repo (99.7% Java, 0.3% HTML) (Android App),
IITBApp repo (96.3% Python, 3.7% HTML) (Backend)

1. d. What is the activity cycle of a basic Android application? Diagrams / flowcharts preferred.

An activity is the single screen in android. It is like a window or frame of Java. By the help of activity, you can place all your UI components or widgets in a single screen.

The 7 Lifecycle methods of Android activity:

1. onCreate() - called when activity is first created.
2. onStart() - called when activity is becoming visible to the user.
3. onResume() - called when activity will start interacting with the user.
4. onPause() - called when activity is not visible to the user.
5. onStop() - called when activity is no longer visible to the user.
6. onRestart() - called when activity is stopped, prior to start.
7. onDestroy() - called before the activity is destroyed.


Activity launched
onCreate()
onStart()
onRestart()
User navigates to the activity
onResume()
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

1. e. What are 5 different UI elements in an android app? One example is a “TextView”.

In android, UI or input controls are the interactive or View components that are used to design the user interface of an application. Those are TextView, EditText, Buttons, Checkbox, Progressbar, Spinners, etc.

1. f. [BONUS]What are some of the salient features of those languages(part c)? How similar are they to C++?

The Java programming language is easy to learn. Java code is easy to read and write.
Java is similar to C/C++ but it removes the drawbacks and complexities of C/C++ like pointers and multiple inheritance. So if you have a background in C/C++, you will find Java familiar and easy to learn.
Unlike C++ which is semi object-oriented, Java is a fully object-oriented programming language. It has all OOP features such as abstraction, encapsulation, inheritance and polymorphism.
Since Java SE version 8 (JDK 8), Java has been updated with functional programming features like functional interfaces and Lambda Expressions. This increases the flexibility of Java.
Java has automatic garbage collection and a simple memory management model (no pointers like C/C++).
Java code is compiled into bytecode which is highly optimized by the Java compiler, so that the Java virtual machine (JVM) can execute Java applications at full speed. In addition, compute-intensive code can be re-written in native code and interfaced with Java platform via Java Native Interface (JNI) thus improving the performance.
The Java platform is designed with multithreading capabilities built into the language. That means you can build applications with many concurrent threads of activity, resulting in highly interactive and responsive applications.
Java code is compiled into intermediate format (bytecode), which can be executed on any system for which Java virtual machine is ported. That means you can write a Java program once and run it on Windows, Mac, Linux or Solaris without re-compiling. Thus the slogan “Write once, run anywhere” of Java.

Kotlin:

Officially released in 2016, Kotlin has attracted a lot of attention in recent years, especially since Google announced its support for Kotlin as an alternative to Java on Android platforms.
Kotlin is a modern, statically-typed programming language that features both object-oriented and functional programming constructs. It targets several platforms, including the JVM, and is fully interoperable with Java. In many ways, Kotlin is what Java might look like if it were designed today.
Salient features:

1.Clean, compact syntax,
2.Single type system (almost)
3.Null safety
4.Functions and functional programming
5.Data classes
6.Extensions
7.Operator overloading
8.Top-level objects and the Singleton pattern

TASK

1. Read about relative and linear layouts and how they are used to design the UI of Apps.

The basic building block for user interface is a View object that is created from the View class and occupies a rectangular area on the screen. Views are the base class for UI components like TextView, Button, EditText etc.
The ViewGroup is a subclass of View. One or more Views can be grouped together into a ViewGroup. A ViewGroup provides the android layout in which we can order the appearance and sequence of views. Examples of ViewGroup are LinearLayout, RelativeLayout etc.

Linear Layout: A layout that arranges its children in a single column or a single row.
Relative Layout: This layout is a view group that displays child views in relative positions.

Android LinearLayout organizes elements along a single line. We can specify whether that line is vertical or horizontal using android:orientation. The orientation is horizontal by default.

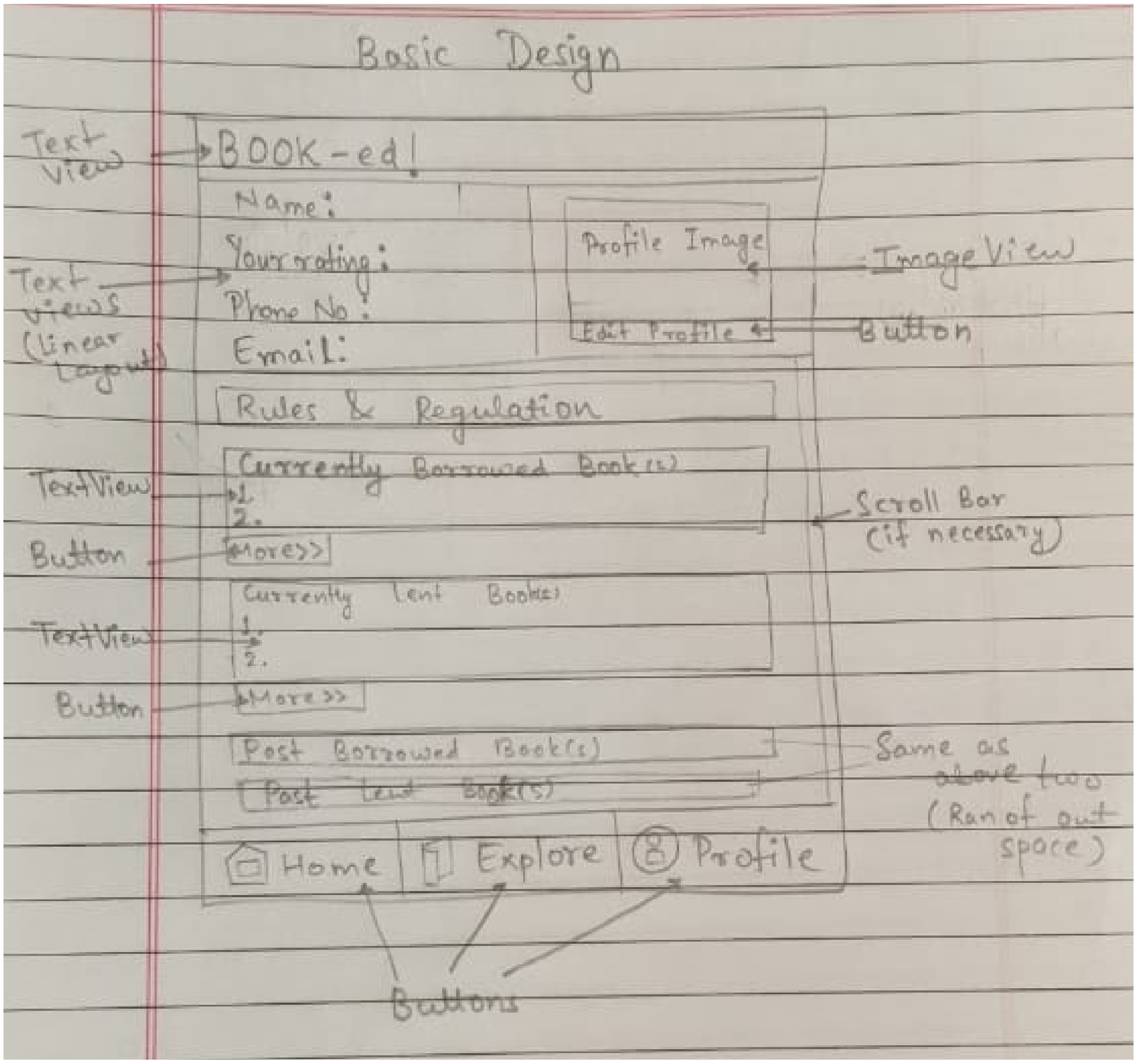
Android RelativeLayout lays out elements based on their relationships with one another, and with the parent container. This is one of the most complicated layouts and we need several properties to actually get the layout we desire. That is, using RelativeLayout we can position a view to be toLeftOf, toRightOf, below or above its siblings. We can also position a view with respect to its parent such as centered horizontally, vertically or both, or aligned with any of the edges of the parent RelativeLayout. If none of these attributes are specified on a child view then the view is by default rendered to the top left position.

2. Now suppose you want to design the landing screen / dashboard of Book-ed!, what do you think should be the various features of that screen?

3. Draw a schematic diagram of the screen. You can do this using a pen and paper or use online android prototyping tools.

4. Now break down your design into various layouts and elements. Clearly mark what is a linear layout, what are the various elements being used in your design. For example if there is a piece of text somewhere on the screen, that part would be the "TextView".

Home screen / Dashboard would contain summarized profile, status of borrowed and lent books which is what the user wishes to see first.
He/she can navigate to the 'Explore Books' page (or change their profile)...



<u>OR</u>

The bottom three buttons can be in linear layout (horizontal).

Currently Borrowed Books … Past Lent Books (the entire thing) can be in another linear layout (vertical).

Instead of using ‘Empty Activity’ we can use ‘Bottom Navigation Activity’ when selecting a new project. (I still used ‘Empty Activity’)

5. [BONUS] Install “Android Studio” on your laptop with all the necessary requirements and run the starter app on your android device.
6. [BONUS] Write the code for the dashboard which you just designed / prototyped.

I did the tutorial and it can be found in my git repo as MyFirstApp.
I forked and cloned the repo of InstiApp, and tried building it but had some build issues.
I designed my prototype dashboard and can be found in my repo as MyDashboard. I imagined better but wasn’t able to enhance it.

References:

https://www.android.com/
https://en.wikipedia.org/wiki/Android_(operating_system)
https://github.com/pulsejet/iitb-app-angular
https://github.com/wncc/IITBapp
https://github.com/wncc/InstiApp
https://www.geeksforgeeks.org/top-programming-languages-for-android-app-development/
https://www.javatpoint.com/android-life-cycle-of-activity
https://www.edureka.co/blog/activity-lifecycle/
https://www.geeksforgeeks.org/android-app-development-fundamentals-for-beginners/
https://www.tutlane.com/tutorial/android/android-ui-controls-textview-edittext-radio-button-checkbox
https://www.codejava.net/java-core/features-of-the-java-programming-language
https://www.javaworld.com/article/3396141/why-kotlin-eight-features-that-could-convince-java-developers-to-switch.html
https://www.journaldev.com/9495/android-layout-linearlayout-relativelayout
https://www.youtube.com/watch?v=dFlPARW5IX8

Devansh Jain
190100044