Prasad Khurd

EasyBizz



An Android Application

* Software Components used:

1. Android Studio
2. InteliiJ IDEA
3. Firebase Database
4. Android Emulator (Google Nexus API 30)
5. Adobe Xd (Vector Graphics)
6. JSON Prettier.
7. Java JDK Version 1.8
8. Android Software Dependencies
9. Database Software Dependencies

* Hardware Components Used:

1. Physical Devices (Oppo A7, Samsung galaxy A9(2018), Samsung J7 Neo)
2. Wi-fi Card for checking network connectivity

**Software Components**

Android Studio: Android Studio is the official integrated development environment for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development

IntelliJ IDEA: IntelliJ IDEA is an integrated development environment written in Java for developing computer software.

Firebase Database: Firebase is a database platform developed by Google for creating scalable databases for both mobile and web applications. Firebase can be used for both Android as well as iOS operating systems.  It was originally an independent company founded in 2011. In 2014, Google acquired the platform and it is now their flagship offering for app development

Android Emulator: Android Emulator is a virtual Android device that can be installed on remote operating system (like Windows 10, Mac OS X)

It can be used to run Android apps without any physical Android device.

Adobe Xd: Adobe XD is a vector-based user experience design tool for web apps and mobile apps, developed and published by Adobe Inc. It is available for macOS and Windows, although there are versions for iOS and Android to help preview the result of work directly on mobile devices.

Java JDK: The Java Development Kit is a distribution of Java Technology by Oracle Corporation. It implements the Java Language Specification and the Java Virtual Machine Specification and provides the Standard Edition of the Java Application Programming Interface. In this project, we’ve used a Java JDK which comes pre-built with Android Studio.

Android Software Dependencies: Android software dependencies are set of software codes that are pre- written by particular organization.

We just have to integrate them into our Android project.

For this Particular project, we’ve used following software dependencies:

* 'androidx.appcompat:appcompat:1.4.0'
* 'com.google.android.material:material:1.4.0'
* 'androidx.constraintlayout:constraintlayout:2.1.2'
* 'androidx.lifecycle:lifecycle-livedata-ktx:2.4.0'
* 'androidx.lifecycle:lifecycle-viewmodel-ktx:2.4.0'
* x'androidx.legacy:legacy-support-v4:1.0.0'
* 'com.google.firebase:firebase-storage:20.0.0'
* 'com.squareup.picasso:picasso:2.71828'
* testImplementation 'junit:junit:4.+'

Database Software Dependencies: Database software dependencies are set of database software codes that are pre- written by particular organization, here Firebase Inc.

We just have to integrate them into our Android project.

For this Particular project, we’ve used following software dependencies:

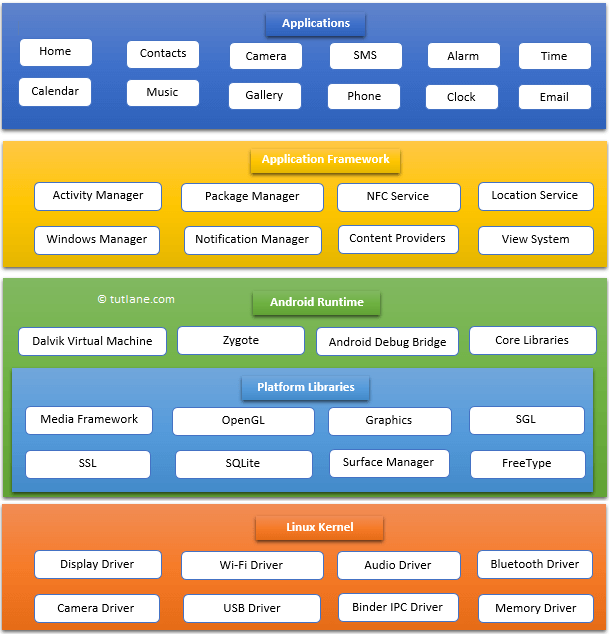
* 'com.google.firebase:firebase-auth:21.0.1'
* 'com.google.firebase:firebase-database:20.0.3'
* ‘com.firebaseui:firebase-ui-database:6.0.2'

**Hardware Components**

Physical Device: A physical Android device is a actual Android device which we can look and feel at. For this particular project, we’ve used OPPO A7, Samsung Galaxy A9 (2018), Samsung J7 Neo as our physical devices across.

Wi-fi Card: In this project, we’ve used wi-fi card for checking network connectivity of our android application while software testing. Airtel Wi-fi card was used for this project.

Android Component Diagram:



Individual Software components of Android OS:

* Service

A [Service](https://developer.android.com/reference/android/app/Service) is an [application component](https://developer.android.com/guide/components/fundamentals#Components) that can perform long-running operations in the background. It does not provide a user interface. Once started, a service might continue running for some time, even after the user switches to another application. Additionally, a component can bind to a service to interact with it and even perform inter-process communication (IPC). For example, a service can handle network transactions, play music, perform file I/O, or interact with a content provider, all from the background.

* Broadcast Receiver:

Android apps can send or receive broadcast messages from the Android system and other Android apps, similar to the [publish-subscribe](https://en.wikipedia.org/wiki/Publish%E2%80%93subscribe_pattern) design pattern. These broadcasts are sent when an event of interest occurs. For example, the Android system sends broadcasts when various system events occur, such as when the system boots up or the device starts charging. Apps can also send custom broadcasts, for example, to notify other apps of something that they might be interested in (for example, some new data has been downloaded).

Apps can register to receive specific broadcasts. When a broadcast is sent, the system automatically routes broadcasts to apps that have subscribed to receive that particular type of broadcast.

Generally speaking, broadcasts can be used as a messaging system across apps and outside of the normal user flow. However, you must be careful not to abuse the opportunity to respond to broadcasts and run jobs in the background that can contribute to a slow system performance

* Activity:

The [Activity](https://developer.android.com/reference/android/app/Activity) class is a crucial component of an Android app, and the way activities are launched and put together is a fundamental part of the platform's application model. Unlike programming paradigms in which apps are launched with a main() method, the Android system initiates code in an [Activity](https://developer.android.com/reference/android/app/Activity) instance by invoking specific call-back methods that correspond to specific stages of its lifecycle. This document introduces the concept of activities, and then provides some lightweight guidance about how to work with them. For additional information about best practices in architecting your app, see [Guide to App Architecture](https://developer.android.com/topic/libraries/architecture/guide).