# ABSTRACT

This work deals with the development of an android-based application that integrates the videos and images as in the Instagram app. The Application allows us to view the images and videos posted by the other users and interact with them. This android-based application is created using xml and java.

# CONTENTS

Chapter

1. Introduction
   1. Introduction
   2. Major Development Platforms
   3. Cost of Developing a Mobile App
   4. The Mobile App Development Process
2. Overview of the project
   1. Overview
   2. Objectives
   3. Scope and Limitation
3. Testing and Maintenance
   1. Introduction to Programming Languages
      1. Java
   2. Introduction to IDE’s
      1. Android Studio
      2. Android Software Development Kit (SDK)
   3. Test Plan and Test Activities
4. System Requirement Specification
   1. Software Requirements Specification Document
      1. Data Requirements
      2. Functional Requirements
      3. Performance Requirements
5. Implementation
6. Snapshots

Conclusion and Future Scope

CHAPTER 1

INTRODUCTION

1.1 Introduction

Mobile application development is the process to making software for smartphones and digital assistants, most commonly for Android and iOS. The software can be preinstalled on the device, downloaded from a mobile app store or accessed through a mobile web browser. The programming and mark-up languages used for this kind of software development include Java, Swift, C# and HTML5.

Mobile app development is rapidly growing. From retail, telecommunications and e- commerce to insurance, healthcare and government, organizations across industries must meet user expectations for real-time, convenient ways to conduct transactions and access information. Today, mobile devices and the mobile applications that unlock their value—are the most popular way for people and businesses to connect to the internet. To stay relevant, responsive and successful, organizations need to develop the mobile applications that their customers, partners and employee’s demand.

Yet mobile application development might seem daunting. Once you’ve selected the OS platform or platforms, you need to overcome the limitations of mobile devices and usher your app all the way past the potential hurdles of distribution. Fortunately, by following a few basic guidelines and best practices, you can streamline your application development journey.

Mobile apps are majorly developed for 3 Operating System:

* Android
* IOS
* Windows

There are 3 different ways to develop Mobile apps: –

* 1st Party Native App development
* Progressive web Application
* Cross-Platform Application

1. 1st Party Native App development:

These types of apps normally run in the native devices, that is, it runs only in the OS that it is specifically designed for it. These apps cannot be used on different devices using a different OS. The apps that are developed for android are normally coded using Java or Kotlin languages. The IDE normally used for android app development is Android Studio which provides all features and the apps that are developed for IOS are generally coded in Swift language or Objective-C. The IDE suggested for IOS App Development is XCode.

1. Progressive web Application:

Progressive web apps are essentially a website which runs locally on your device. The technologies used are Microsoft Blazor, React, Angular JS, Native Script, Iconic. These technologies normally used for web development propose. The apps’ UI is developed the same way as they are developed while developing the website. This category has many ups and downs let’s start with the advantages of Progressive web apps.

1. Cross-Platform Application:

These are frameworks that allow developing total native applications which have access to all the native features of IOS and Android but with the same code base. These apps run on both Android and IOS. So normally the development speeds of these apps are very fast and the maintenance cost is low. The performance speed is comparatively low to 1st party native apps but faster than PWA.

Flutter is Google’s cross-platform solution which uses the language, Dart. The IDE preferred is Android Studio, IntelliJ IDE, and Visual Studio Code. The UI/UX is bespoke and Flutters has to come up with their new libraries whenever Android and IOS comes up with an update to mimic those updates. The community is fast growing.

Major Mobile Development Platforms

* + iOS

iOS is kind of arguably the “big dog” once it comes to major app development platforms,

partially because it had been the platform that finally brought mobile development into the modern day and age by fully reworking the concept of a mobile device and mobile software system. iOS is of course developed by Apple, and it runs exclusively on Apple products. Apple provides iOS developers with many native tools and libraries to develop iOS applications, and, though you do not have to be enforced to use Apple’s development tools to create your apps, you just need to have a mac running OS X to build your application.

* + Android

Android is the other dominant player in this space, it was a bit later participant to the game, first being released in Sept 2008, virtually a year later than iOS but it has managed to achieve a reasonably massive share of the mobile market. Technically, Android is the mobile OS with the largest most dominant share of the market with around 80% share compared to iOS’s 18 % share. Those numbers are a bit deceiving since android may be a fragmented market consisting of the many different devices created by different manufacturers, running completely different versions of the Android OS. Many independent application development teams choose to build their apps for Android first. Why? The vast majority— around 70 percent—of smartphones run Android, and the Google Play Store has fewer restrictions than the Apple App Store. On the other hand, mobile applications developed for iOS have far fewer devices that need support, making optimization simpler. And user retention is typically higher for iOS applications.

Depending on the intended use case and target audience for the mobile application you are developing, you might have other considerations. For example, if you’re designing an app for your organization’s employees, you’ll need to support the platforms they use, which may mean developing cross-platform. Apps that work for both Android and iOS. Or if you’re building a mobile application for your customers and you know the majority of them use iPhones, then developing iOS applications should be a top priority. Additional considerations when developing your mobile applications include monetization strategies and anticipated user behavior, which can be influenced by geographical and cultural factors.

Cost of developing a Mobile App

The cost of developing an app can range from almost nothing to millions of dollars -- it all depends on the type of app and its intended use. Following is a breakdown of the ranges you can expect to pay for building a mobile app:

* No-code app builders: A developer doesn't need to know how to code if the app has basic feature requirements. Free tools such as GoodBarber, Appery.io, Shoutem, Appy Pie and Build Fire offer the freedom to build apps without learning Swift or other programming languages. Although these tools are limited in their functionality and can't be used to create a game with no-code apps, the no-code approach will meet most organization's needs.
* Enterprise apps: The concept of Citizen Developer, where anyone can build a mobile app, is exploding with tools such as Amazon's HoneyCode, Mendix and Microsoft Power Suite. These devices offer drag-and-drop interfaces that can connect to data sources and manage content flow. The price is typically tied to a monthly subscription of less than $50.
* Mobile optimized website: Although it's most practical to build websites for both desktop and mobile devices, the website content management tool you're using will likely have plugins you can buy for less than $100 to optimize your website for mobile devices.
* Complex apps: An app that requires features, such as 3D, gaming or sophisticated artificial intelligence (AI), will likely need to be developed as a native app. The cost for a complex app can typically be $250,000 or more. The price is directly related to the scarcity of mobile developers.

The Mobile App Development Process

The following steps should help you develop the framework for building an app.

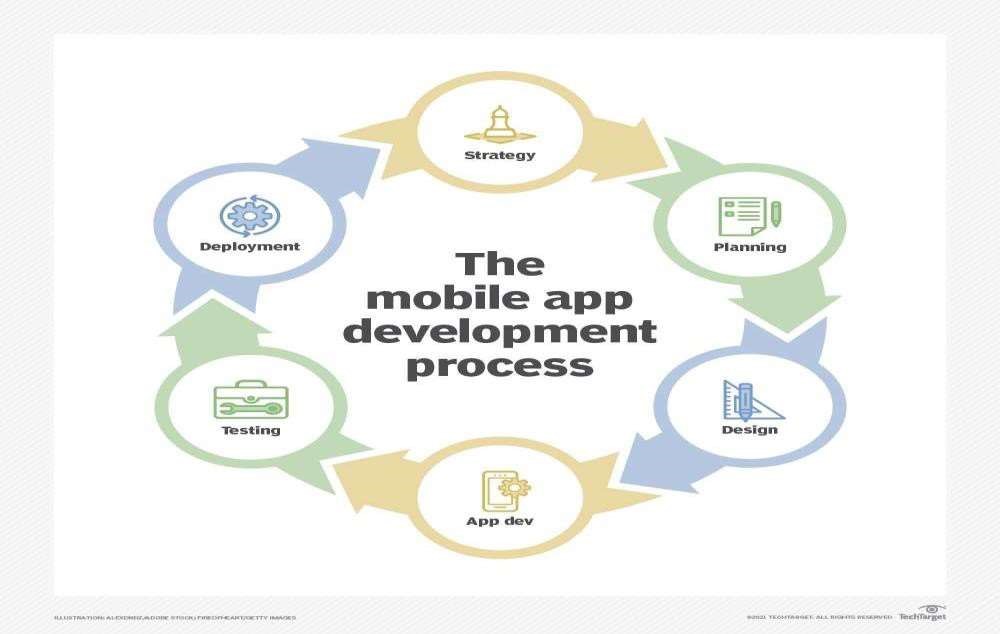


Figure – 1.2 – Mobile App Development Process

Steps in Mobile App Development Process

* Step 1- Mobile App Development Strategy

We need to always define a strategy for our mobile app development process to help unfold your end-product into a successful reality. In case of our Instagram News Feed App, Our strategy is about providing the users to engage with the app.

* Step 2- Analysis and Planning for Mobile App and Development Process Proper planning and analysis have been done to implement the features that is planned.

Our main goal is to meet the maximum requirements of today's technology.

* Step 3- UI / UX Design in Mobile App Development Process

In this step we have designed the android structure layout and added the required designs in case of frontend and backend as well.

|  |  |
| --- | --- |
|  | Step 4 - App Development  The most integral part of this phase is always the planning. The three integral parts in developing a mobile app are back-end/server technology, API(s), and the mobile app front-end. |
|  | Step 5 - Testing  We first need to develop test cases that cover all the components of app testing to ensure a complete QA process for our app. Test cases include test steps, software quality evaluation which will have recorded findings, and re-testing which is done by tracking fixes. |
|  | Step 6 - Deployment & Support  We have to submit your native apps before their release to the Apple App Store for iOS apps and Google Play for Android apps, you must have a developer account as part of the Apple App Store and Google Play Store. |

CHAPTER 2

**OVERVIEW OF PROJECT**

* 1. Overview

Instagram News Feed App is implemented in java using the software Android Studio. The Main aim of this project is to develop an application through which users can easily see videos and images. This application is designed for users who expect privacy and user-friendly app design.

* 1. Objectives

The core objective of this project is to provide an android application that will enable users to engage with other users via the Instagram News feed. Another objective is to provide users a platform for those who want to be able to watch videos and images of other users locally.

* 1. Scope and Limitation

The users who use the app can engage in a healthy way by viewing the images and videos posted by the other users. Like and comment button is also provided so that the user can interact with the images and the user who posted them.

* 1. Android Platform to Meet the Requirements

Nowadays people have been moved so much into the modern technology that they really want a portable living environment along with intelligent objects which contain powerful infrastructure with the most desired features. Thus, android mobile applications have become very popular among the smartphone users. Among these Android Applications, a fun-filling game will provide a great way to pass time.

CHAPTER 3

**TESTING AND MAINTENANCE**

3.1 Introduction to Programming Languages

**3.1.1 Java**

As the project is developing an Android Application, the default programming language is Java. All Android applications are built using Java in Android Studio or Eclipse or both. Java is a popular and widely used language throughout the world. As mentioned in, Java is one of the powerful programming languages like C, C++. developed by Sun Microsystems which has many powerful features as described below. After the development of C, C++, Java has come into evolution by addressing their drawbacks. It is one of the open-source projects that could be easily installed in our machine. The language is also easy to learn, understand and implement. Java is used in various kinds of applications like Web, Desktop, Mobile, and Big Data. Many powerful features are supported by Java including various libraries, application services, graphics library for 2D/3D applications. The language is flexible enough to maintain code complexity, test, implementation, integration and support. Apart from these, there are other key features which make Java more special. It is object oriented programming language, one of the important hierarchies in the programming languages which is used to implement real time applications, it provides for code reusability, it has a platform independence feature including any virtual machines(Write Once Read Everywhere), as in no need to write the 20 code for different OS as the Java Compliers convert the java source files to bytecode and this could be interpreted by any machine and the actual code is compiled irrespective of any machine, OS. It is more secured as the compilers are designed efficiently to figure out any kind of errors.

3.2 **Introduction to IDE’S**

* + 1. **Android Studio**

Android Studio is exclusively designed for.8. developing Android applications. It consists of all Android SDK tools to design, develop, maintain, test, debug and publish our app. The IDE is designed very efficiently which makes the developer’s job easy. It also supports the IntelliJ IDE, the main idea behind this IDE is that it automatically senses the variables, methods, classes, built-in functions or it could be anything else when we press the first letter of it. Say, suppose we declared few variables or methods that starts with an ‘S’, it automatically senses everything that starts with an ‘S’ and makes suggestions. It also supports Git as a version control system to maintain the app changes and push them into GitHub. All java files, layout files (for design) are integrated into a single project easily. After the completion of project, the whole application could be put as an .APK (Android Package) file, in which we can run that APK file in any device and use the application. Other main tools include Android SDK, ADB, and Gradle Build.

* + 1. Android Software Development Kit (SDK):

One of the main tools used in developing android applications, as it packages many core features into one SDK and it can be used in the application easily. This helps us to avoid writing lot of code, and building applications faster.

* + 1. Android Debug Bridge (ADB):

Android SDK uses ADB tool as a connection device which allows us to connect the Android Devices or Emulator with the machine via USB. After developing or while developing applications, we can connect with the device to check how the application runs. Later, we can debug and run the applications.

3.3 Introduction to Tools and Technologies used for this implementation

* + 1. Gradle Build:

Gradle Scripts are the recent feature that is added to Android Studio. It is basically an automated build system which is used to automate the various phases involved in designing an application that includes design, development, test, debug, and publish. We need to configure the project and modules by mentioning all the supported jar files, SDK’s, version name, level, compiled SDK version, build tools version. to ensure that the developed app is compatible with the testing device/emulator. Gradle is also similar to Ant and Maven which helps in maintaining java projects (repositories).

* + 1. Android Device Monitor:

If we want to access all the hidden files that are generated when we run the application, we can use the monitor. We can select any project and explore the files that are related to that project. But, as they are hidden files, we need root permissions to access them. Suppose, if we run the app in device, we need to root the device and run commands in adb shell to get permissions.

* + 1. SDK Manager:

It is one of the main tools to maintain the updates of all the installed components required to run the project. It also notifies us when the project is not compatible with device or any other compatibility issues and to download any component that is required.

* + 1. AVD Manager:

It is used to create virtual devices of any desired API level to support higher level SDKs in case our device does not support. Using emulators to test the application is difficult as it might be little slower when compared to real device.

CHAPTER 4

**SYSTEM REQUIREMENT SPECIFICATION**

4.1 Software Requirements Specification Document

* + 1. Data Requirements

The set of data that is involved in any project is defined using data requirements. For this project, the main data required is the login information to register the application and the item’s information. Without this information the application cannot process the transaction.

* + 1. Functional Requirements

Functional requirements are properties that must exist in the final system. For any mobile application, we need to download the application from the play store. The application could be either free or paid depending upon the store or merchant. To use the application, the user needs to register and login to the application after installing by providing login information. Once, he or she logins into the application, they can use all the features.

* + 1. Performance Requirements

Response time, scalability, platform dependencies, tolerance are the performance requirements that should be considered when developing any system. The application or system should be able to respond quickly when the user interacts with the application. The application should be developed in such a way that it should be scalable enough to accept new features when we want to expand the application complexity. The application 5 should run in all the specified software and hardware requirements from the design phase of the project. Also, the tolerance rate (fault tolerance) of the application should be at a higher level in case of network issues, connectivity issues, and when the application crashes or stops. It should be able to deliver the information about any of those issues to the user when the system is no longer able to provide results when the user wants.

* + 1. System Requirements

The application should be installed into a device, system or any machine in such a way that it should have basic requirements like supporting software and hardware of the device, accessing in-built software, say camera for mobile device, internet permissions, and potential security issues such as virus or any malware detection.

* + 1. Testing and Maintainability Requirements

The application should be able to meet all the possible good and bad test cases under a test environment. Application should be developed in such a way that it does not have any issues or crashes when the user is using the application. It should be able to extend itself when we expand the code or implement any new functions to the existing application.

* 1. Validation

Validating any application is an important criterion before releasing the application to the users. If there is no validation, the information entered by users may be redundant, formatted inappropriately and cannot be maintained. For example, we can validate mobile number in a way that it should use only digits and letters. Suppose, if the validation is not done, there are chances for the user to enter a wrong phone number and save it. In case of any emergency issues, the authorized person cannot contact the respective person. Similarly, validations for all the fields that are used to save information in any application are highly necessary. In this application, I have done several validations in the Login Page and Home Page. In the Login Page, I have validated all the login information that is required for the user to sign up for the first time. Fields like username, email, password, mobile number are validated appropriately by displaying error messages. The username should not contain any digits, password should be minimum of six letters, the email should be a valid address and if the email id is already registered, an error message is shown saying that email id exists. The mobile number should contain only digits, when scanning the item, scan operation should be done properly to fetch the barcode number, if not a message is shown. When searching the item using any name, it should not contain any digits in the search field.

* 1. Android Development Tools

Android is the significant platform for open-source mobile applications. It uses Linux kernel based as default operating system; using Android SDK to build user application in Java. Android is more flexible when it can be developed in both Mac and Windows systems. Using the add-on Android SDK, we not only can develop but also can run and test the application on both of the emulators and real Android devices. Below Table describes the development tools in android platform.

|  |  |
| --- | --- |
| Specification | Usage |
| Minimum Android API level | API 21 |
| Area of usage | Anywhere |
| Hardware | Laptop or Desktop with 4Gb RAM  Android device (for testing) |
| Languages | 1. Java      1. XML (eXtensible Markup Language) |
| Android Development | 1. Eclipse IDE |

|  |
| --- |
| CHAPTER 5    IMPLEMENTATION    MODULES.XML  <?xml version="1.0" encoding="UTF-8"?>  <project version="4">  <component name="ProjectModuleManager">  <modules>  <module fileurl="file://$PROJECT\_DIR$/.idea/InstagramVideoFeedClone main.iml" filepath="$PROJECT\_DIR$/.idea/InstagramVideoFeedClone-main.iml" /> </modules>  </component>  </project>              WORKSPACE.XML  <?xml version="1.0" encoding="UTF-8"?>  <project version="4">  <component name="AutoImportSettings">  <option name="autoReloadType" value="NONE" />  </component>  <component name="ChangeListManager">  <list default="true" id="b5a89ce2-bf4f-4985-b1b4-354a3184e583" name="Default Changelist" comment="" />  <option name="SHOW\_DIALOG" value="false" />  <option name="HIGHLIGHT\_CONFLICTS" value="true" />  <option name="HIGHLIGHT\_NON\_ACTIVE\_CHANGELIST" value="false" />  <option name="LAST\_RESOLUTION" value="IGNORE" />  </component>  <component name="ProjectId" id="2C62rXwSNP8btuucURDSO0zdD7u" />  <component name="ProjectViewState">  <option name="hideEmptyMiddlePackages" value="true" />  <option name="showLibraryContents" value="true" />  </component>  <component name="PropertiesComponent">  <property name="RunOnceActivity.OpenProjectViewOnStart" value="true" />  <property name="RunOnceActivity.ShowReadmeOnStart" value="true" />  <property name="RunOnceActivity.cidr.known.project.marker" value="true" />  <property name="cidr.known.project.marker" value="true" />  <property name="last\_opened\_file\_path" value="$PROJECT\_DIR$" />  </component>  <component name="SpellCheckerSettings" RuntimeDictionaries="0" Folders="0" CustomDictionaries="0"  DefaultDictionary="application-level" |
| UseSingleDictionary="true" transferred="true" />  <component name="TaskManager">  <task active="true" id="Default" summary="Default task">  <changelist id="b5a89ce2-bf4f-4985-b1b4-354a3184e583" name="Default Changelist" comment="" />  <created>1658108912791</created>  <option name="number" value="Default" />  <option name="presentableId" value="Default" />  <updated>1658108912791</updated>  </task>  <servers />  </component>  </project>    MAINACTIVITY.JAVA  package com.nehak.instagramfeed import android.content.Intent import android.os.Bundle  import androidx.databinding.DataBindingUtil import androidx.fragment.app.FragmentActivity  import com.nehak.instagramfeed.autoPlay.VideoPreLoadingService import com.nehak.instagramfeed.dataModels.FeedItem  import com.nehak.instagramfeed.databinding.ActivityMainBinding import com.nehak.instagramfeed.other.Constants import java.util.\*  class MainActivity : FragmentActivity() {  override fun onCreate(savedInstanceState: Bundle?) { super.onCreate(savedInstanceState)  DataBindingUtil.setContentView<ActivityMainBinding>(this, R.layout.activity\_main) startPreCachingOfVideo()  }  /\*\*   * This method will start service to preCache videos from remoteUrl in to Cache Directory * So the Player will not reload videos from server if they are already loaded in cache   \*/  fun startPreCachingOfVideo() {  val preloadingServiceIntent = Intent(this, VideoPreLoadingService::class.java) preloadingServiceIntent.putStringArrayListExtra(  Constants.VIDEO\_LIST,  Constants.videoList.toStringArray()  )  startService(preloadingServiceIntent) }  } |

|  |
| --- |
| private fun <E> ArrayList<E>.toStringArray(): ArrayList<String> {    val arr = ArrayList<String>();    forEach {  arr.add((it as FeedItem).downloadUrl!!);  } return arr;    } |

## MAINACTIVITYTEST.JAVA

|  |
| --- |
| package com.nehak.instagramfeed  import androidx.test.espresso.Espresso.onView  import androidx.test.espresso.assertion.ViewAssertions.matches import androidx.test.espresso.matcher.ViewMatchers.isDisplayed import androidx.test.espresso.matcher.ViewMatchers.withId import androidx.test.ext.junit.rules.ActivityScenarioRule import androidx.test.ext.junit.runners.AndroidJUnit4 import androidx.test.filters.LargeTest import org.junit.Rule import org.junit.Test  import org.junit.runner.RunWith  @LargeTest  @RunWith(AndroidJUnit4::class) public class MainActivityTest {  @Rule @JvmField  public var activityScenarioRule = ActivityScenarioRule(MainActivity::class.java)  @Test  fun FeedListFragment\_isVisible\_True() {  onView(withId(R.id.feed\_list\_fragment)).check(matches(isDisplayed()));  }  } |

|  |  |
| --- | --- |
| MAINAPPLICATION.JAVA  package com.nehak.instagramfeed import android.app.Application  import com.google.android.exoplayer2.database.ExoDatabaseProvider  import com.google.android.exoplayer2.upstream.cache.LeastRecentlyUsedCacheEvictor import com.google.android.exoplayer2.upstream.cache.SimpleCache import com.nehak.instagramfeed.other.Constants.Companion.simpleCache import java.io.File  class MainApplication : Application() { override fun onCreate() { super.onCreate() setUpForPreCaching()  }  fun setUpForPreCaching() {    val exoPlayerCacheSize = 50 \* 1024 \* 1024.toLong()// Set the size of cache for video var leastRecentlyUsedCacheEvictor: LeastRecentlyUsedCacheEvictor? = null var exoDatabaseProvider: ExoDatabaseProvider? = null if (leastRecentlyUsedCacheEvictor == null) {  leastRecentlyUsedCacheEvictor = LeastRecentlyUsedCacheEvictor(exoPlayerCacheSize)  }  if (exoDatabaseProvider != null) {  exoDatabaseProvider = ExoDatabaseProvider(this)  }  if (simpleCache == null) {  val cache: File = File(getCacheDir(), "Video\_Cache") if (!cache.exists()) { cache.mkdirs()  }  simpleCache =  SimpleCache(cache, leastRecentlyUsedCacheEvictor, exoDatabaseProvider)  }  }  }    FEEDLISTFRAGMENTTEST.JAVA  package com.nehak.instagramfeed  import androidx.fragment.app.testing.FragmentScenario import androidx.fragment.app.testing.launchFragmentInContainer import androidx.test.espresso.Espresso.onView  import androidx.test.espresso.assertion.ViewAssertions.matches import androidx.test.espresso.matcher.ViewMatchers.\* import androidx.test.ext.junit.runners.AndroidJUnit4 import androidx.test.filters.LargeTest  import com.nehak.instagramfeed.feedUI.FeedListFragment import org.junit.Before import org.junit.Test | |
| import org.junit.runner.RunWith    @LargeTest  @RunWith(AndroidJUnit4::class)  class FeedListFragmentTest {      lateinit var scenario : FragmentScenario<FeedListFragment>;    @Before fun launch(){  //launch the fragment before each test  val scenario = launchFragmentInContainer<FeedListFragment>()  }    @Test  fun is\_RootView\_Visible\_shouldBe\_True() { onView(withId(R.id.root\_view\_feed\_fragment)).check(matches(isDisplayed()))    }    @Test  fun is\_RecyclerView\_Visible\_shouldBe\_True() {  onView(withId(R.id.recycler\_view)).check(matches(isDisplayed()))  }    @Test  fun is\_ListItem\_Visible\_shouldBe\_True() { onView(withId(R.id.recycler\_view)).check(matches(hasDescendant(withId(R.id.feedThumbnailView))))  } | |

## EXAMPLEINSTRUMENTEDTEST.JAVA

|  |
| --- |
| package com.nehak.instagramfeed    import androidx.test.platform.app.InstrumentationRegistry import androidx.test.ext.junit.runners.AndroidJUnit4 import org.junit.Test import org.junit.runner.RunWith    import org.junit.Assert.\*    /\*\*   * Instrumented test, which will execute on an Android device.   \*   * See [testing documentation](http://d.android.com/tools/testing).   \*/  @RunWith(AndroidJUnit4::class) class ExampleInstrumentedTest {  @Test fun useAppContext() {  // Context of the app under test.  val appContext = InstrumentationRegistry.getInstrumentation().targetContext assertEquals("com.nehak.instagramfeed", appContext.packageName)  }  } |

## EXAMPLEUNITTEST.JAVA

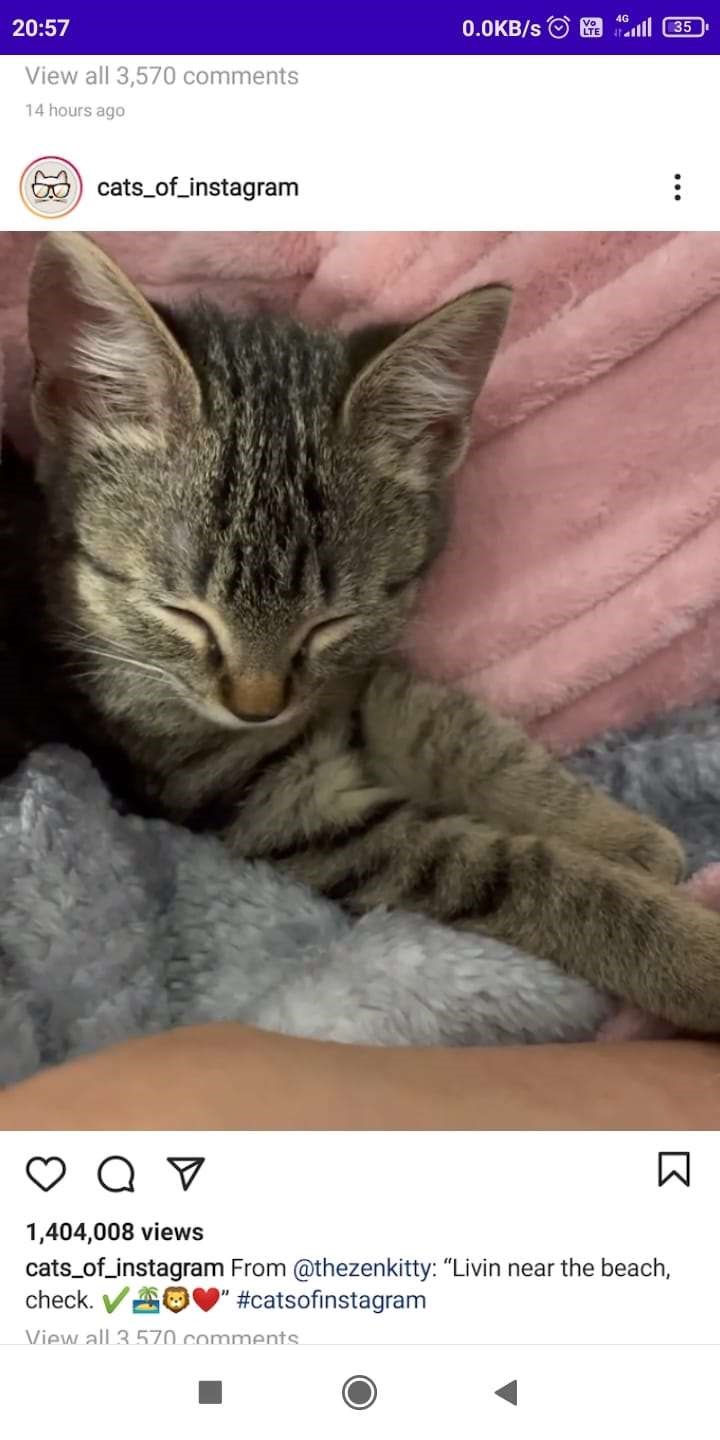
|  |
| --- |
| package com.nehak.instagramfeed    import org.junit.Test    import org.junit.Assert.\*    /\*\*   * Example local unit test, which will execute on the development machine (host).   \*   * See [testing documentation](http://d.android.com/tools/testing).   \*/  class ExampleUnitTest {  @Test  fun addition\_isCorrect() { assertEquals(4, 2 + 2)  }  } |

CHAPTER 6

SNAPSHOTS







CONCLUSION AND FUTURE SCOPE

The package was designed in such a way that future modifications can be done easily. The following conclusions can be deduced from the development of the project. Automation of the entire system improves the efficiency. It provides a friendly graphical user interface which proves to be better when compared to the existing system. It gives appropriate access to the authorized users depending on their permissions. It effectively overcomes the delay in communications. Updating of information becomes so easier. System security, data security and reliability are the striking features. The System has adequate scope for modification in future if it is necessary. This application avoids the manual work and the problems concern with it. Hence the application can be updated regularly and maintained with continuous releases. There is also scope for upgradation to premium features and monetization of the application.