Student-Teacher Interaction System

Project report submitted in partial fulfillment of the requirement
for the degree of Bachelor of Technology

in

Computer Science and Engineering/Information Technology

By

Ishaan Vikram (141228)

Parashiv Sihaniya (141233)

Under the supervision of

Dr. Rajinder Sandhu

To

Department of Computer Science & Engineering and Information
Technology

**Jaypee University of Information Technology, Waknaghat,
Solan-173234, Himachal Pradesh**

CANDIDATES' DECLARATION

We hereby declare that the work presented in this report entitled **"Student-Teacher Interaction System"** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering** submitted in the department of Computer Science and Engineering**,** Jaypee University of Information Technology, Waknaghat, is an authentic record of our own work carried out over a period from January 2018 to May 2018 under the supervision of **Dr. Rajinder Sandhu**, Assistant Professor, Department of Computer Science and Engineering.

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

Ishaan Vikram (141228)

Parashiv Sihaniya (141233)

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

Dr. Rajinder Sandhu

Assistant Professor

Department of Computer Science and Engineering

Dated:

ACKNOWLEDGEMENT

We are grateful and indebted to Dr. Rajinder Sandhu, Assistant Professor, Department of Computer Science and Engineering, and Mr. Puneet Kumar Jain, for their help and advice in the completion of this project report.

We also express our deep sense of gratitude and appreciation to our guide for his constant supervision, inspiration and encouragement right from the beginning of this project.

We also want to thank our parents and friends for their immense support and confidence upon us. We deem it a pleasant duty to place on record our sincere and heartfelt gratitude to our project guide for his long sightedness, wisdom and co-operation which helped us in tackling crucial aspects of the project in a very logical and practical way.

Ishaan Vikram (141228)

Parashiv Sihaniya (141233)

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

This project aimed at developing a mobile Android based application based on student-teacher interaction since smartphone applications are finding use into almost every pathway in our fast paced lives. As the proficiency and achievement of an academic class is fairly reliant on the level and extent of student-teacher interaction in class, this project report underlines how our mobile application can viably contribute in expanding communication and give the important criticism to make the present and future classes better ones.

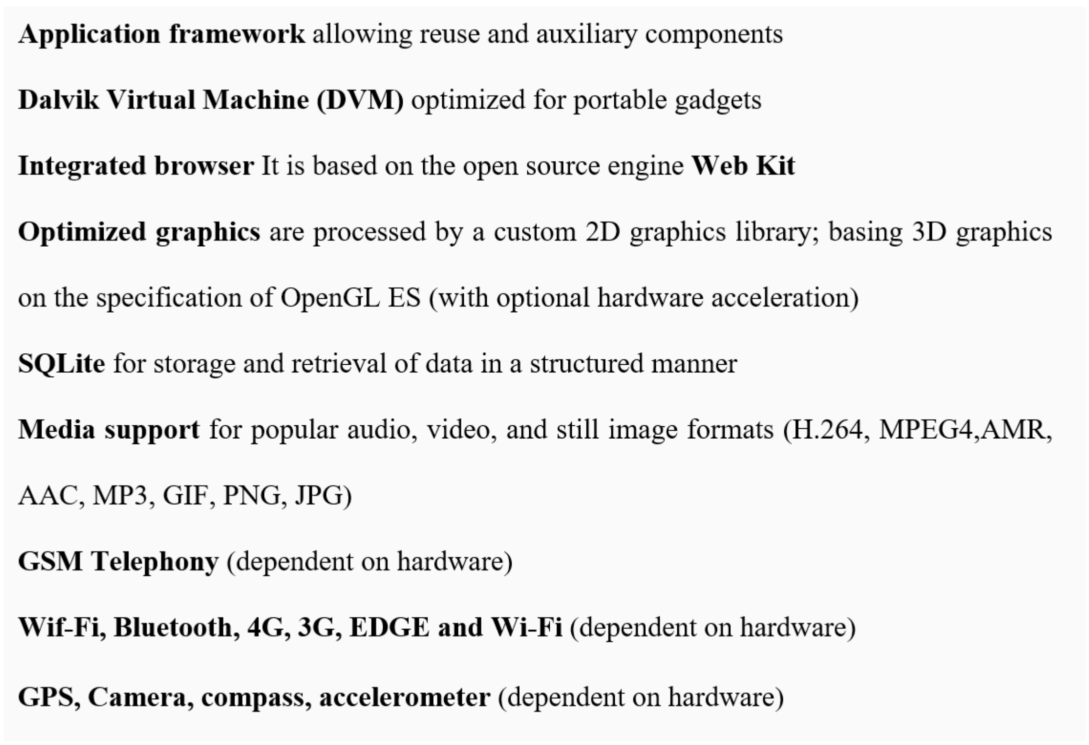
Today, the impact that the communication amongst teachers and students has on learning and educating is an irrefutable issue. Deprived of interaction, it is hard for learners to be motivated to continue their courses, and the learning gets obtuse. This discussion of interacting in an e-learning environment is more complicated than conventional education. Some ways to surmount this challenge of interaction involve using customized techniques of presentation speed, style, the graphics, and survey content. Another good way is the formation of student associations and discussion group

CHAPTER 1
INTRODUCTION

1.1 Introduction

Android is an open source mobile operating system (OS) written primarily in Java and based on the Linux Kernel and outlined fundamentally for touchscreen cell phones, for example, tablets and cell phones. It is currently developed by Google. According to Google's official Android documents, "Android's user interface is largely based on using touch gestures loosely corresponding to real-world actions such as tapping, swiping and/or pinching, to work on-screen interactable objects, with a virtual keyboard for text input. Google has further launched and developed Android TV for televisions, and Android Wear for wrist watches, each with a specialized UIs." Several Android variants are also used on gaming consoles, digital cameras, notebooks, and many other electronics.

Android Components



Application framework allowing reuse and auxiliary components

Dalvik Virtual Machine (DVM) optimized for portable gadgets

Integrated browser It is based on the open source engine **Web Kit**

Optimized graphics are processed by a custom 2D graphics library; basing 3D graphics on the specification of OpenGL ES (with optional hardware acceleration)

SQLite for storage and retrieval of data in a structured manner

Media support for popular audio, video, and still image formats (H.264, MPEG4,AMR, AAC, MP3, GIF, PNG, JPG)

GSM Telephony (dependent on hardware)

Wif-Fi, Bluetooth, 4G, 3G, EDGE and Wi-Fi (dependent on hardware)

GPS, Camera, compass, accelerometer (dependent on hardware)

Figure 1.1 Android Components

1.2 Problem Statement

- An Android application similar to Student Teacher Interaction System already exists in the market.
- Many institutes use an internal portal for interacting between students and faculty, to remain abreast about various on-going events.
- The apps provide many usable functionalities. However, they have certain drawbacks as well:
- The organization of content is not proper and consequently tough to comprehend.
- A large amount of probing is needed.
- Notifications are not sent appropriately, the student has to browse through the app manually to know about upcoming events and new announcements.
- Students from one class can see and alter forums and gatherings and of some different class, which can bring about vagueness of displayed data and make disarray.
- We strived to create an app that would provide a wholesome experience to the user, be it student or faculty.

1.3 Objectives

The target of the design of a new system is to computerize the present technique of supervision and monitoring the data about the students' points of interest and to lessen the overhead of supervising reports for every notification being created.

The proposed system will be responsible for storing the data on a focal server while allowing clients to fetch that data from their own smartphones by making use of the installed android app. A database will be present on the server and an enhanced UI on every customer machine i.e. on the STIS application installed on the user smartphone. The created application will be utilized by understudies, instructors, guardians and administrator.

1.4 Methodology

With overall increase in smart phones usage, the mobile app development market is growsing rapidly. Developing an app that would have a long life span is becoming a challenging task. Hence, it becomes practically compulsory for app development companies to adopt the constantly shifting industry trend in order to gain positive results.

These days, companies use a agile methodology which is most proficiently utilized by software and mobile development organizations accommodating in streamlining the basic development process and make it speedier. The agile approach for building up a mobile app for the most part consists of flexible and adaptable planning, risk managing, customer involvement and continuous evaluation.

This procedure causes improvement group to outline a last item correctly according to the desire of their client. The key advances engaged with a spry improvement process include:

1. Initiation of the Project: Initiation includes setting up the plan, determining the business needs, solidifying the specifications, assembling the group, and getting ready for architecture modeling initially.

2. Development Iterations: This dreary stage includes dynamic customer cooperation, thorough and collective improvement by experts, including new highlights upon prerequisite, corroborative and investigative testing, and interior sending of programming.

3. Release of the product: This stage includes last system testing, last user acknowledgment testing, and sending of system into creation and general and refreshed conveyance of software which meets the changing needs of the customer. Discharge stage enables the end user to audit the software and send the criticism and in addition ask for more highlights if required.

4. Production: This phase involves maintaining the system regularly and identifying defects, enhancing the system stability as well as system performance.

Agile Software Development offers a number of benefits:

- Agile development methodology permits numerous degrees for changes all through the entire life cycle of application improvement
- It augments design sensitivity, sustainability in user experience and app usage.
- Every step needs testing to ensure the product is seamless, enabling to launch the product in a short period of time.
- Tenders comfortable and flexible work conditions to members of the team.
- It uses a reliable work approach.
- Well-organized and swift application process empowers increasing the performance and quality of the project.

- With the continuous communication of the developers, testers and clients, it boosts the transparency and flexibility of the process.

1.5 Organization

Chapter 1: Highlights and underlines the concepts behind the student-teacher interaction system. In this chapter, the introduction to the android based service is covered. The key focus defining the problem statement and specifying the objectives of the project.

Chapter 2: Presents review from journals, research papers and books in deatil. In this chapter, research papers excerpts on research and development of student management systems are given.

Chapter 3: The different stages involved in the the development have been discussed here.

Chapter 4: The methods used for app testing and the performance analysis is shown here.

Chapter 5: Presents concluding remarks about the project and possible scope of the future work to augment the current work with improved efficacy.

CHAPTER 2
LITERATURE SURVEY

According to David Lamb writing for ‘Academic Writing Tutor’, “A literature survey is a text of a scholarly paper, which includes the current knowledge including substantive findings, as well as theoretical and methodological contributions to a particular topic. Literature reviews are secondary sources, and do not report new or original experimental work. Most often associated with academic-oriented literature, such reviews are found in academic journals, and are not to be confused with book reviews that may also appear in the same publication.” The objective of this literature review is to locate the relevant literature about Student-Teacher Interaction and their systems with the end goal of background studying, and to outline the current work and distinguish the gap in present researches.

2.1 Research Papers

2.1.1 “The Development and Design of the Student Management System Based on the Network Environment”

Authors: Zhi -gang YUE,You-wei JIN
The paper talks about the strategy for information management in advanced education. Quoting the summary, “Based on a complete examination and enquiry on the scholar administration in advanced education, the models of the students' administration information were developed by embracing the propelled information innovation, and build the student administration information stage.” Lastly, the crucial strategy plus innovation in completing the administration stage are offered.

2.1.2 “Android-based Attendance Management System”

Authors: Siti Aisah Mohd Noor, Norliza Zaini, Nabilah Hamzah
This papers outlines a technique of placing students’ attendance through usage of an app made for the “Android” platform has been presented in the paper. Once installed, the list of students can be fetched using a nominated web server.

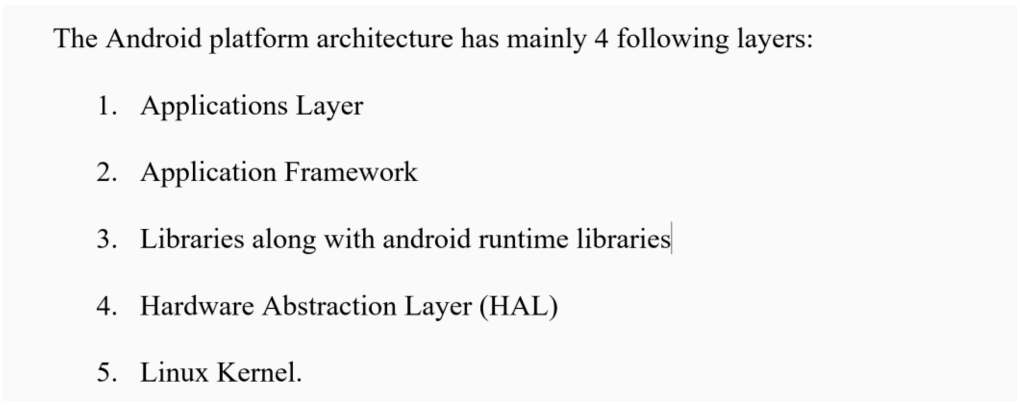
2.1.3 “The Designment of Student Information Management System Based on B/S”

Authors: Jin Mei-shan, Qiu Chang-li, Li Jing
This paper utilizes the B/S structure to outline the student information administration system, and clarifies the system plan standard, system structure and design, the capacity module of information framework as indicated by current college student information administration needs. It gives due for information of substantial figure of students while giving intelligent students administration stage.

2.1.4 “Research and Implementation of Web Services in Android Network Communication”

Authors: Yang Shulin, Hu Jieping.
The paper incorporates blending of Web Services including the way cell phones will advance the improvement of versatile applications. Citing the paper’s summary, “Based on examination and research of the Volley, Ksoap2 and Java Web Services, through the execution of the Http Stack interface and the development of JSON Object Request to acknowledge bolster for Web Services. The plan utilizes JSON configuration to exchange information, bolster S SL/TLS convention demands, custom parameter, sets or gets the demand header.” The above plan has great similarity, utlity, ease of use, and is reasonable for Android stage.

2.2 Architecture of Android



The Android platform architecture has mainly 4 following layers:

1. Applications Layer
2. Application Framework
3. Libraries along with android runtime libraries
4. Hardware Abstraction Layer (HAL)
5. Linux Kernel.

Figure 2.1 Layers of architecture

Each of the layers of the architecture is explained in Figure 2.2 below.

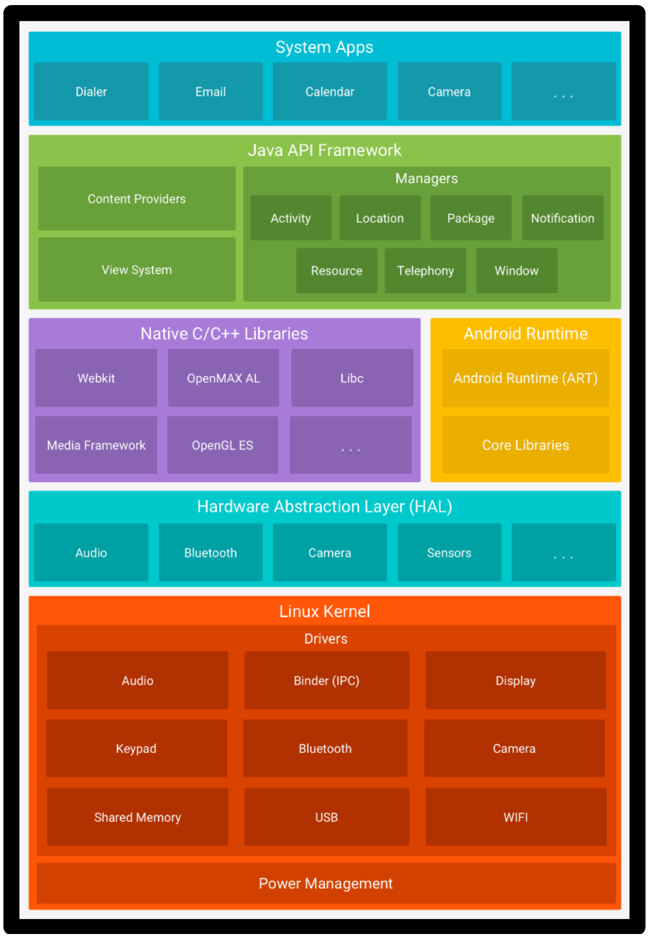


Figure 2.2 Android Architecture

There are 5 components around which an android application revolves. They are:

1. Activities

2. Broadcast Receivers

3. Services

4. Content Providers

5. Intents

2.2.1 Activities

An Activity is an application component providing an interactive screen where users can do something, like making a phone call, taking photos, send emails, or viewing maps. All activities have their own windows to draw their user interface in. This window normally covers the entire screen, but it may be smaller than the screen and float atop other windows.

At the point when an activity is ceased in light of the fact that another activity begins, it is advised of this adjustment in state by means of the activity's lifecycle callback strategies. An activity might obtain several callback methods, owing to changes in its state—whether the system is "stopping it, creating it, resuming it, or destroying it" —and each callback gives the chance to perform specific work suitable to that particular state change.

For example, when stationary, the activity should release somewhat huge objects, like database or network connections. Once the activity recommences, the essential resources and resume actions can be reacquired that were broken up. These state transitions all form the "Activity lifecycle".

2.2.2 Broadcast Receivers

Broadcast Receiver is "a mechanism to transfer events back and forth so that all interested applications can be informed when something happens. There are lots of System events which are broadcast by Android OS, such as connectivity related events, Messaging related events, camera related events, etc.”

Two chief categories of broadcasts that are received are:

Normal broadcasts are completely asynchronous. In a vague request, every collector of the communication are kept running, regularly in the meantime. It is additionally proficient, but it leads to the implication that recipients are not able to utilize the outcome or prematurely finish the included application programming interfaces.

Ordered broadcasts are delivered to a single receiver at a time. When every recipient executes, proliferate an outcome to the following collector, or prematurely end the communication totally with the goal that it won't be passed to different beneficiaries. The order receviers keep running in can be controlled with the priority attribute of the coordinating intent-filter; receivers with a similar priority will be kept running in an order arbitrarily.

Indeed, even on account of typical communicates, the framework may in a few circumstances return to conveying the communicate one collector at any given moment. Specifically, for collectors that may require the formation of a procedure, just a single will be keep running at an opportunity to abstain from over-burdening the framework with new procedures. In this circumstance, be that as it may, the non-requested semantics hold: these beneficiaries still can't return results or prematurely end their communicate.

2.2.3 Intents

Activities, broadcast receivers and services are 3 of the chief application components, and are activated through what are called intents. A facility called Intent messaging is present for delayed run-time binding of components in that application or in different applications. The intent, what we call an Intent object, is a passive data structure which holds a conceptual portrayal of an activity to be performed — or,as it often is in when it comes broadcasts, a depiction of something that has happened and is being declared.

2.2.4 Activity Stack

Activity stack is responsivle for managing the system activities.

At the point when another activity commences, it becomes the running activity and is pushed to the top of the stack -- the past activity reliably stays underneath it in the stack, and does not come to the top again till the new activity is exited.

In the case the Previous button is pressed by the user the subsequent activity of the present stack moves upward and becomes active.

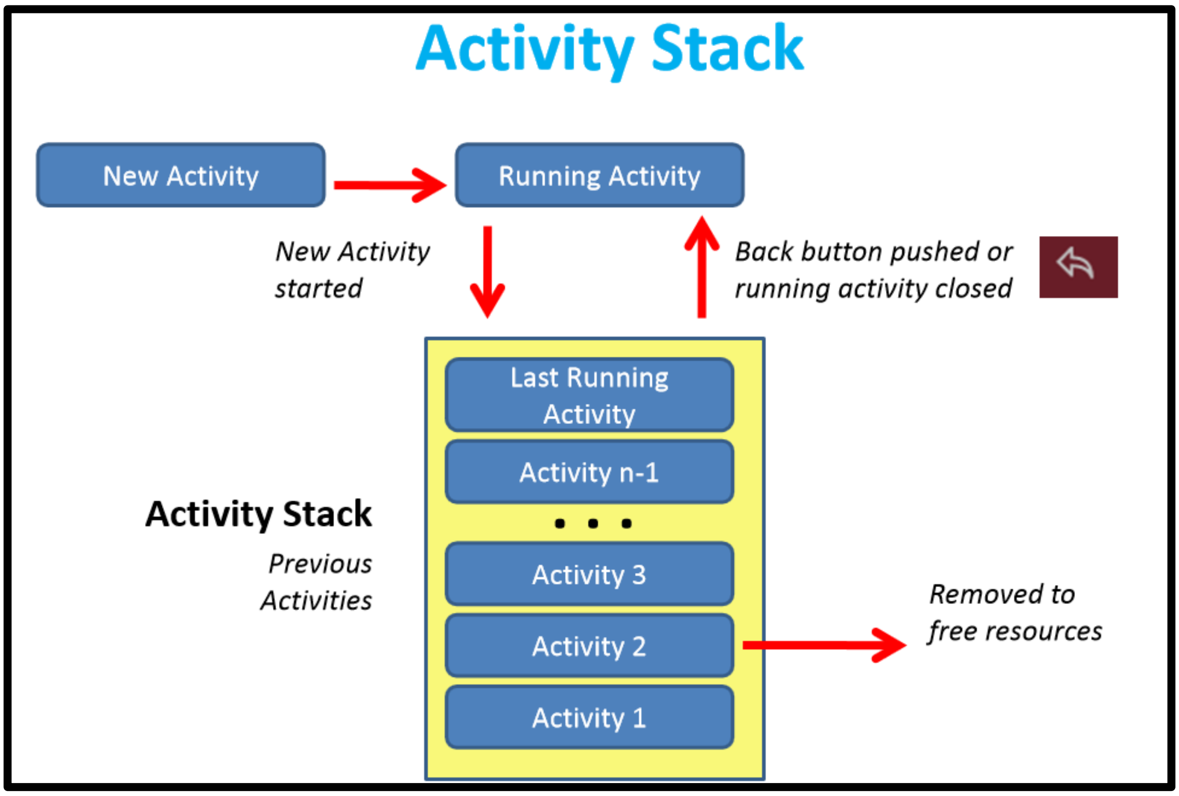


Figure2.3 Activity Stack

2.2.5 Processes and Threads

According to the official Android development website, “When an application segment begins and the application does not have some other segments running, the Android system begins another Linux procedure for the application with a solitary string of execution. As a matter of course, all parts of a similar application keep running in a similar procedure and string (called the "primary" string). In the event that an application segment begins and there as of now exists a procedure for that application (in light of the fact that another part from the application exists), at that point the segment is begun inside that procedure and utilizations a similar string of execution.” Whatver the case may be, distinctive parts in the application can be organized to be kept running in independent procedures. Extra strings can be made for any procedure.

2.2.6 Multi-Tasking

An application for the most part contains different activities. Every movement ought to be composed around a particular sort of activity the user can perform and can begin different activities.

For instance, an email application may have one movement to demonstrate a rundown of new email. At the point when the user chooses an email, another action opens to see that email.

Figure 2.3 explains the Android activity process in the background.

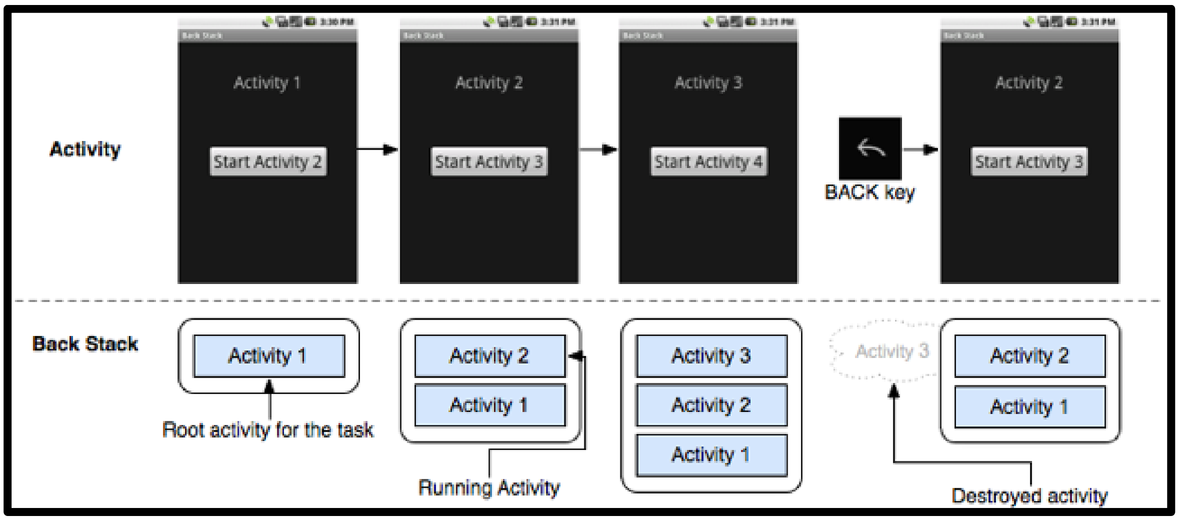


Figure 2.4 Activity in Background

CHAPTER 3
SYSTEM DEVELOPMENT

3.1 System Requirements

These are the software and system necessities of the Android SDK:

3.1.1 Operating System Support

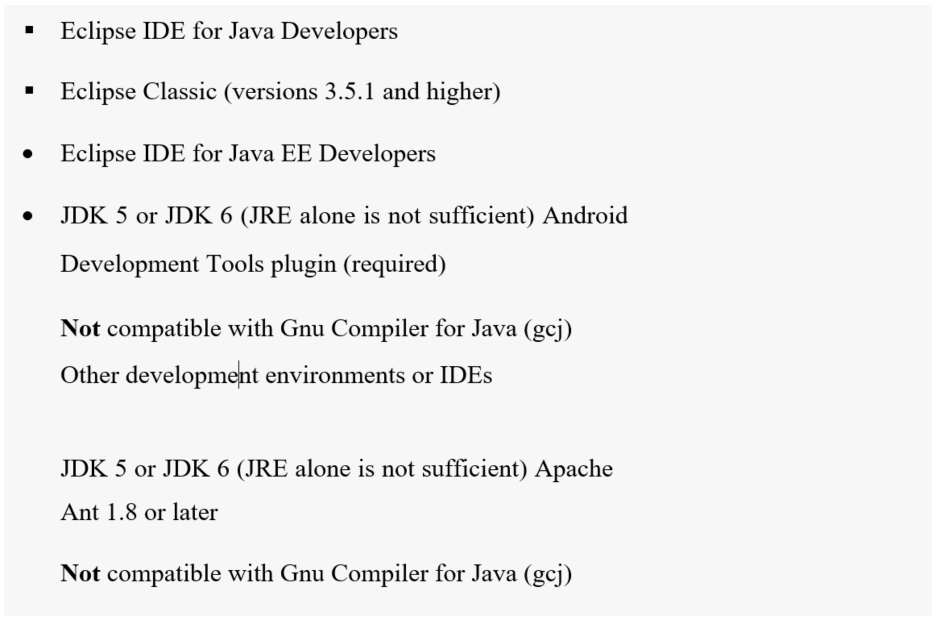
Windows 10 (64-bit), Windows 7 (x64 or x86), Windows Vista (x64 or x86).

Linux (tested on Ubuntu Linux)

GNU “C Library (glibc) 2.7” or later.

On “Ubuntu Linux, version 8.04” or later

For Android applications development, installing one of these packages is recommended:



- Eclipse IDE for Java Developers
- Eclipse Classic (versions 3.5.1 and higher)
- Eclipse IDE for Java EE Developers
- JDK 5 or JDK 6 (JRE alone is not sufficient) Android Development Tools plugin (required)

  Not compatible with Gnu Compiler for Java (gcj)
  Other development environments or IDEs

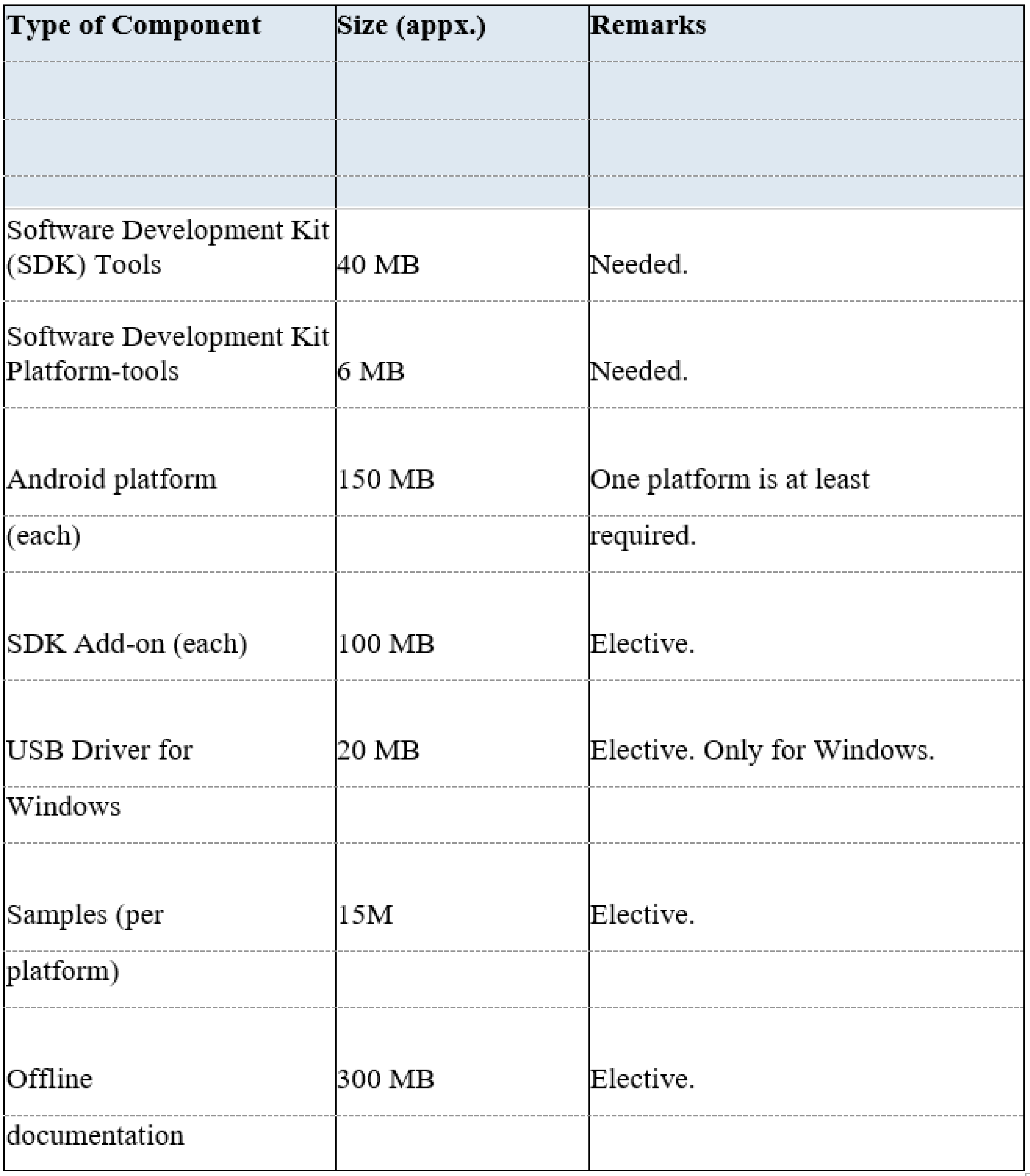
  JDK 5 or JDK 6 (JRE alone is not sufficient) Apache Ant 1.8 or later

  Not compatible with Gnu Compiler for Java (gcj)

Figure 3.a Recommended packages for Android application development

3.1.3 Hardware requirements

For all components that are installed, the Android SDK requires disk storage. Table 3.1 roughly gives an idea of the expected disk-space requirements, depending upon the components which will be used.



Type of Component	Size (appx.)	Remarks
Software Development Kit (SDK) Tools	40 MB	Needed.
Software Development Kit Platform-tools	6 MB	Needed.
Android platform (each)	150 MB	One platform is at least required.
SDK Add-on (each)	100 MB	Elective.
USB Driver for Windows	20 MB	Elective. Only for Windows.
Samples (per platform)	15M	Elective.
Offline documentation	300 MB	Elective.

Table 3.1 Components required by SDK

3.2 Material Design

The following text explains how making revamped changes that follow material design principles will enhance the experience using apps, across Android devices and on other platforms planned for the future.

3.2.1 Discussing material design

According to Google's official support page, "A material representation is the binding together hypothesis of a supported space and an arrangement of movement. The material is grounded in material reality, propelled by the investigation of paper and ink, yet innovatively progressed and open to creative ability and enchantment.

Surfaces and edges of the material give visual prompts that are grounded as a general rule. The utilization of natural material qualities helps clients rapidly comprehend affordances. However the adaptability of the material makes new affordances that supersede those in the physical world, without disrupting the guidelines of physical science.

The basics of light, surface, and development are vital to passing on how protests move, connect, and exist in space and in connection to each other." Sensible lighting demonstrates creases, isolates space, and shows moving parts.

3.3 Working of the app

3.3.1 Use Case Diagram

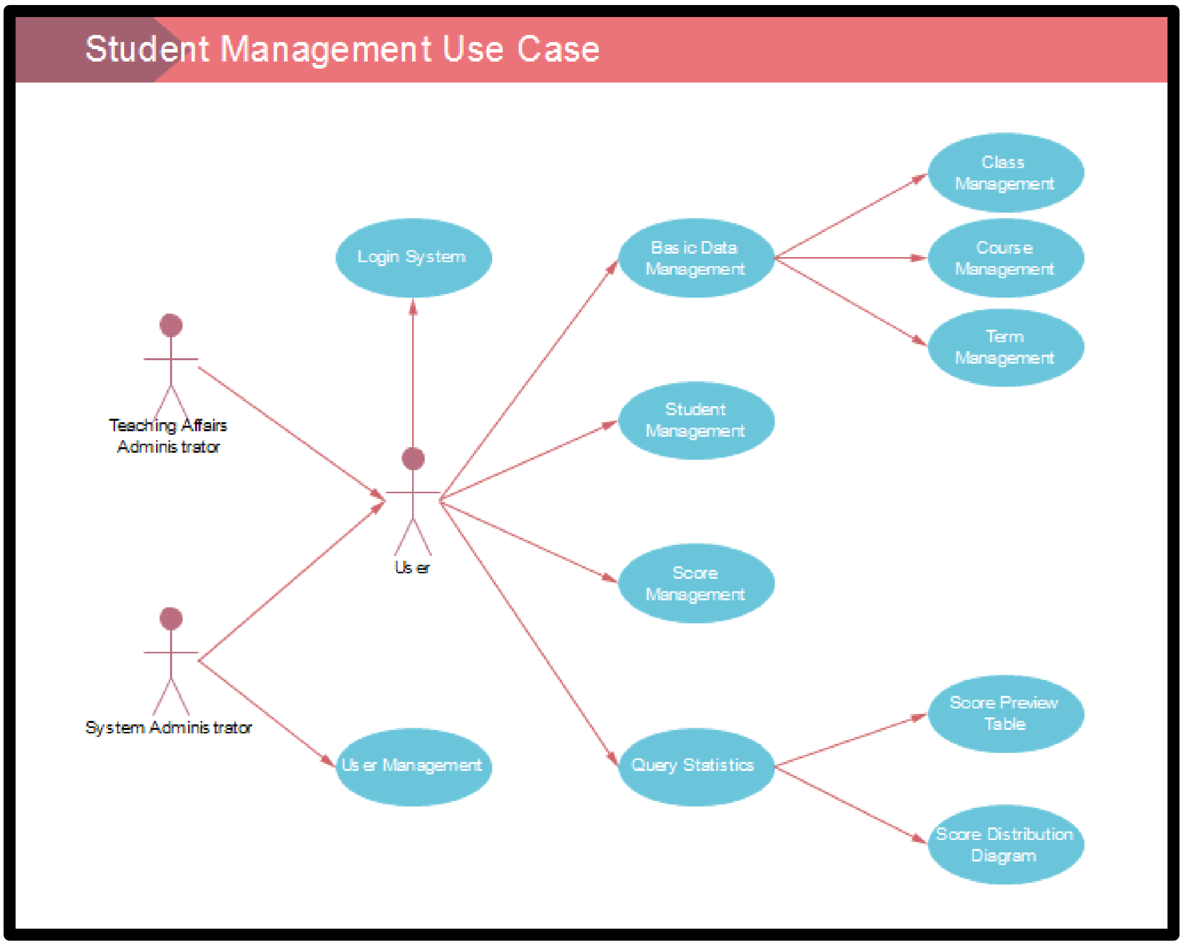


Figure 3.1 Use Case Diagram

3.3.2 Data Flow Diagram

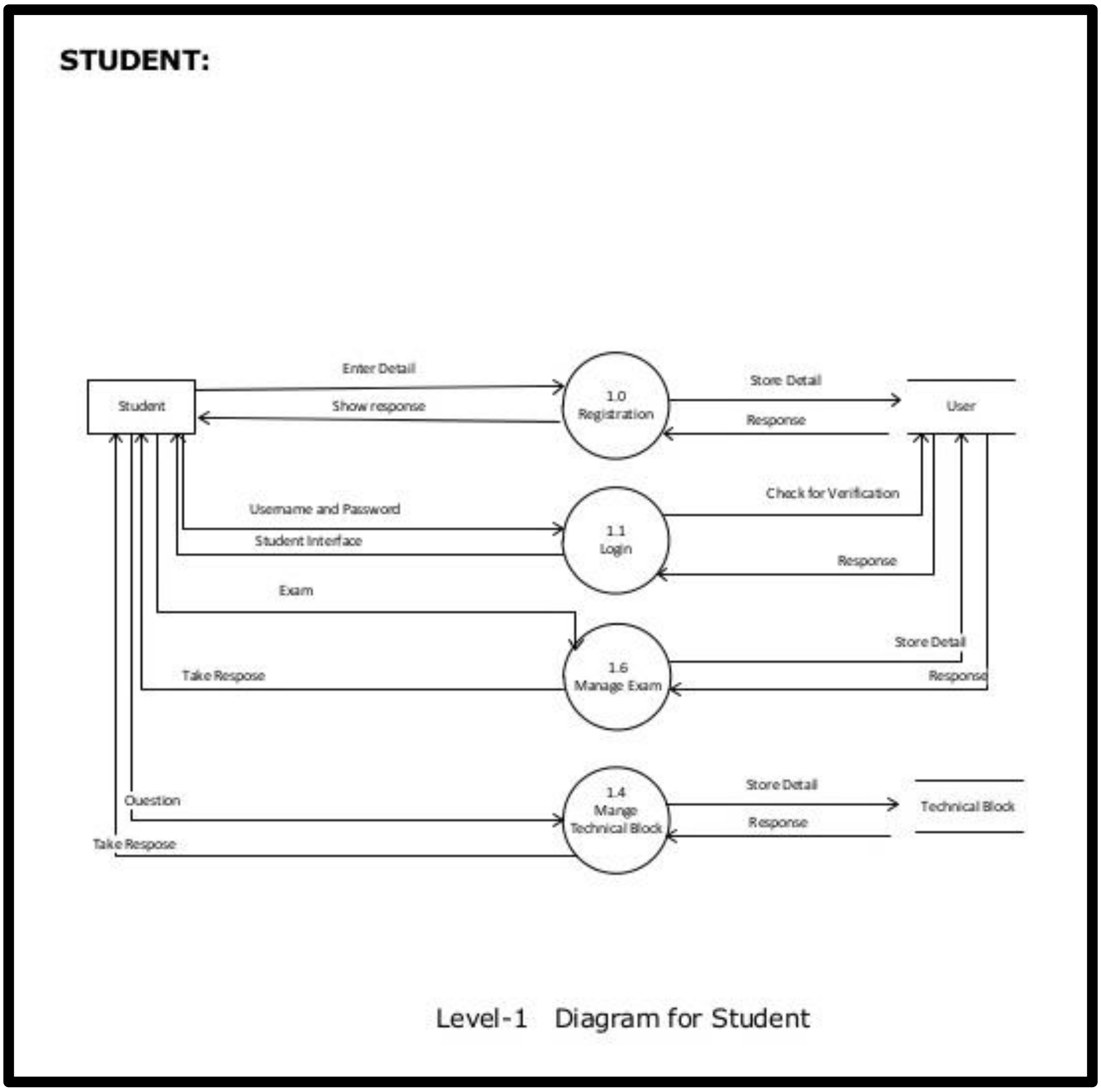


Figure 3.2 Level-1 DFD (Student)

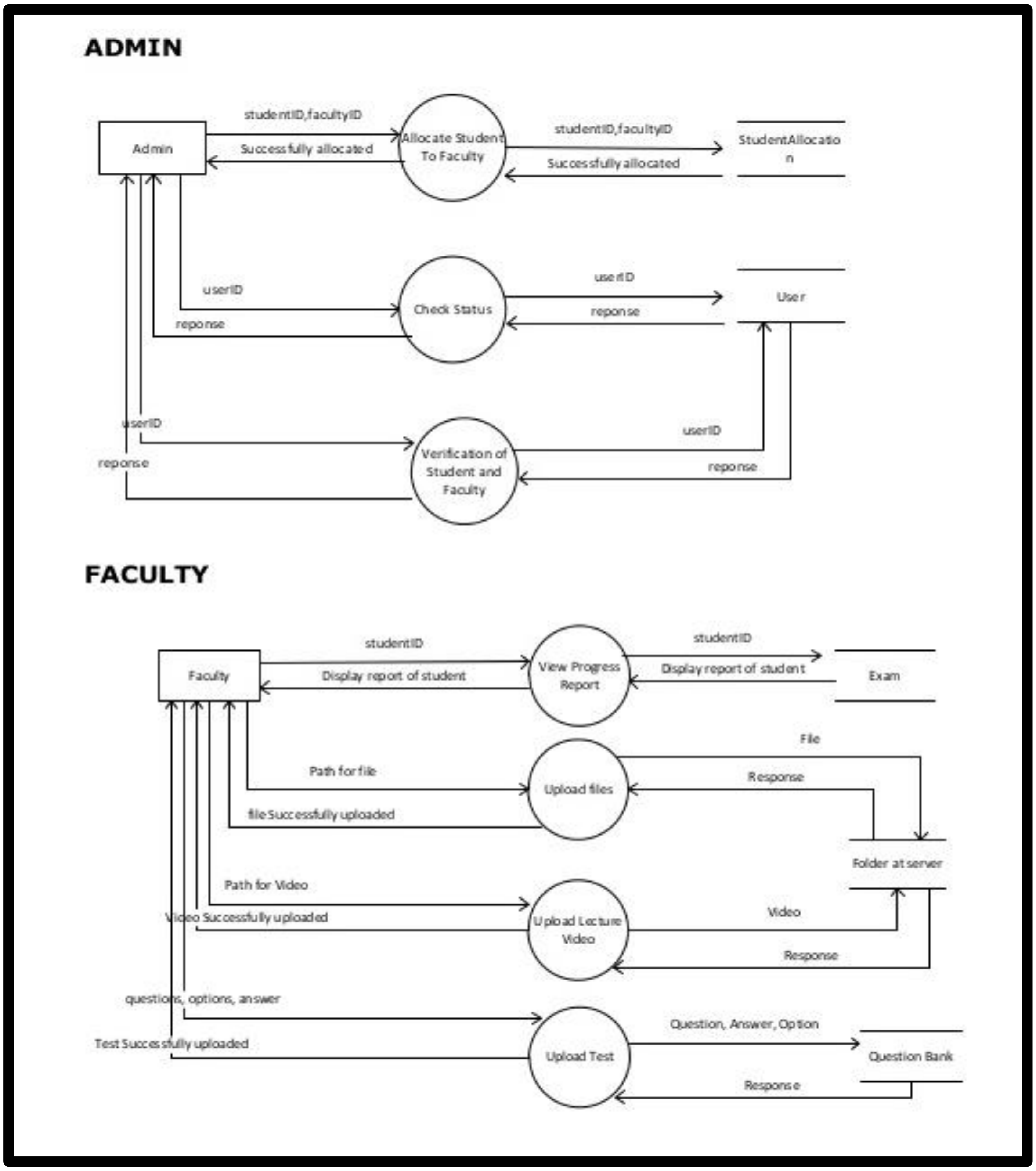


Figure 3.3 Level-2 DFD (Admin and Faculty)

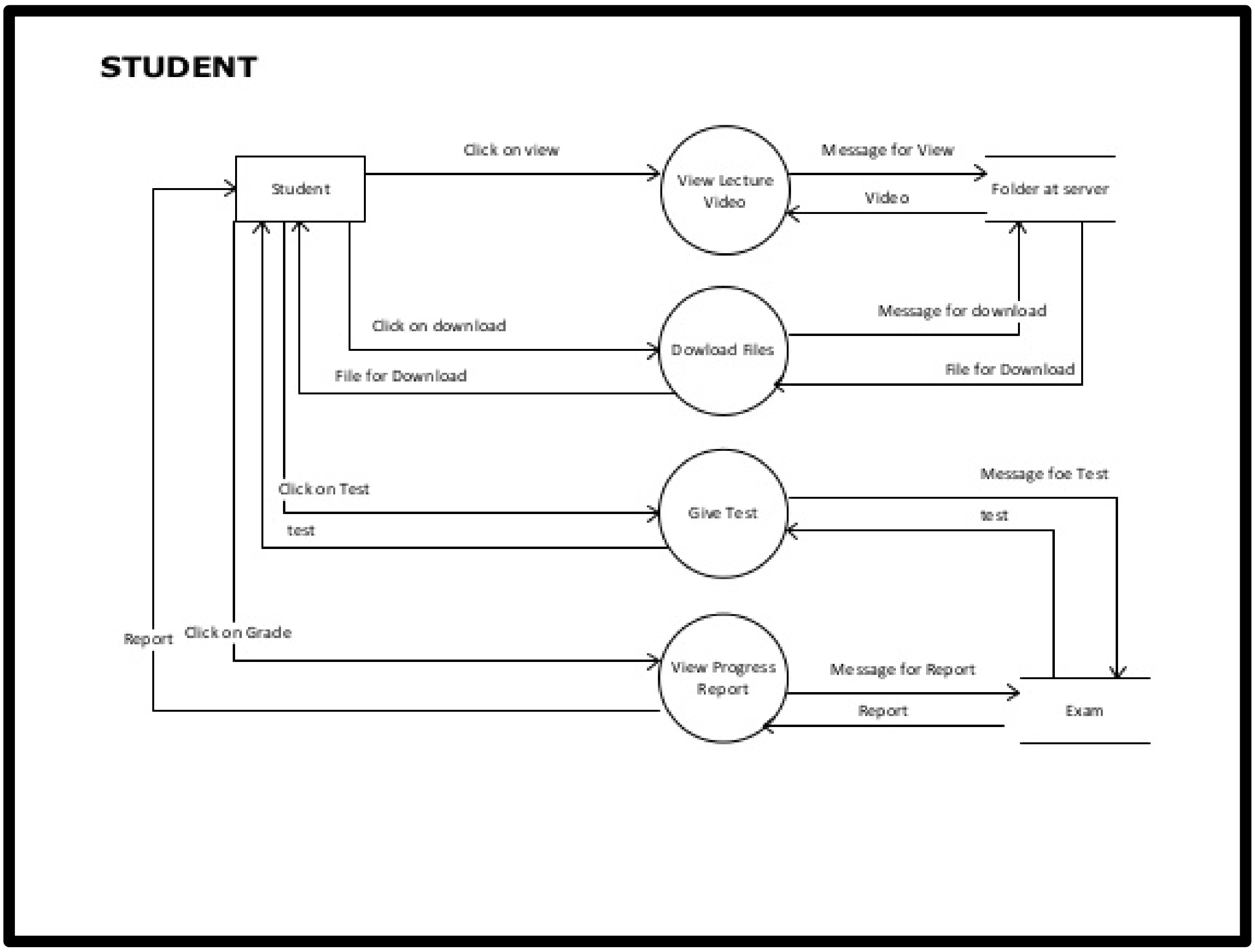


Figure 3.4 Level-2 DFD (Student)

3.3.3 Activity Diagram

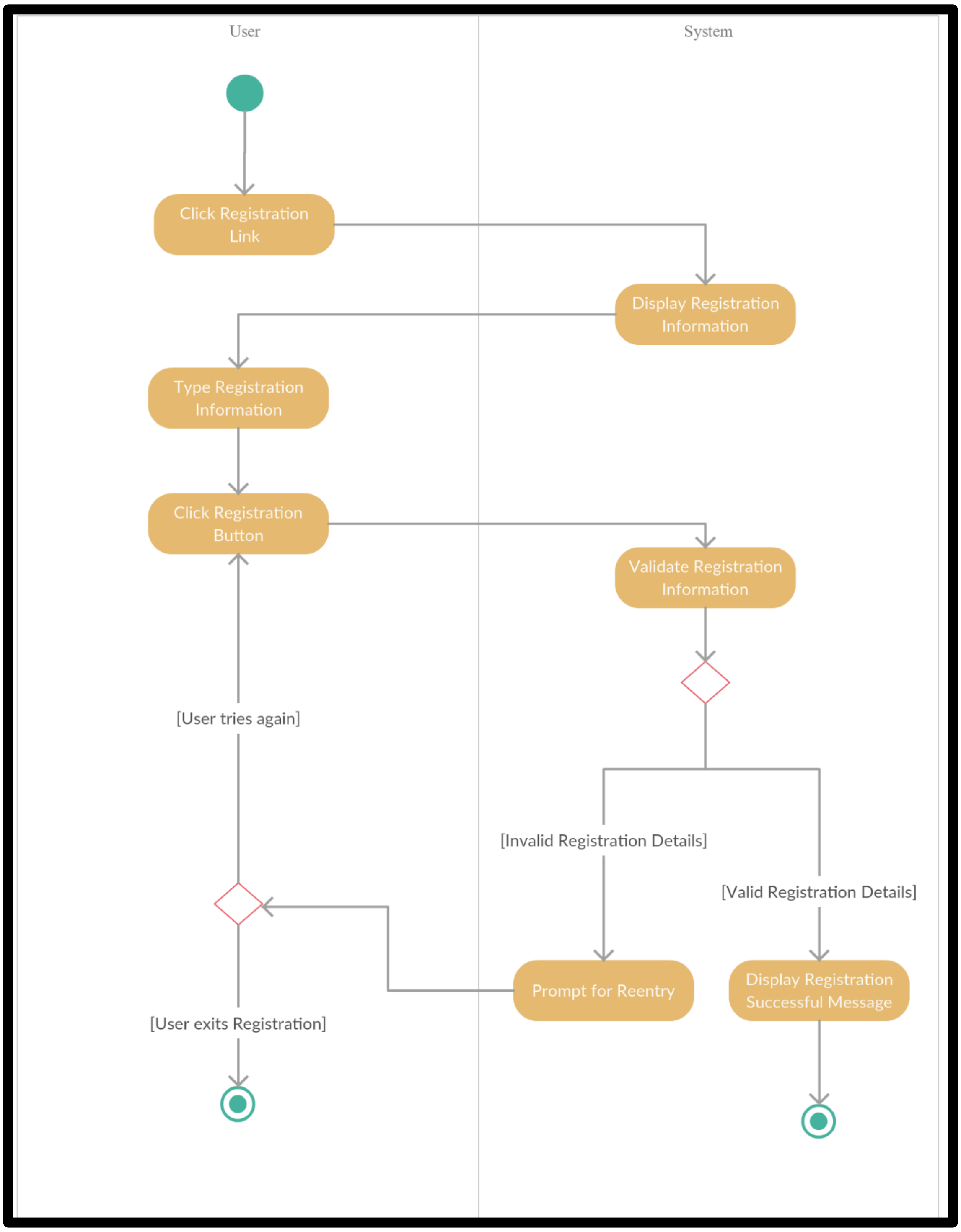

User
System
Click Registration Link
Display Registration Information
Type Registration Information
Click Registration Button
Validate Registration Information
[User tries again]
[Invalid Registration Details]
[Valid Registration Details]
Prompt for Reentry
Display Registration Successful Message
[User exits Registration]

Figure 3.5 Activity Diagram (Registration)

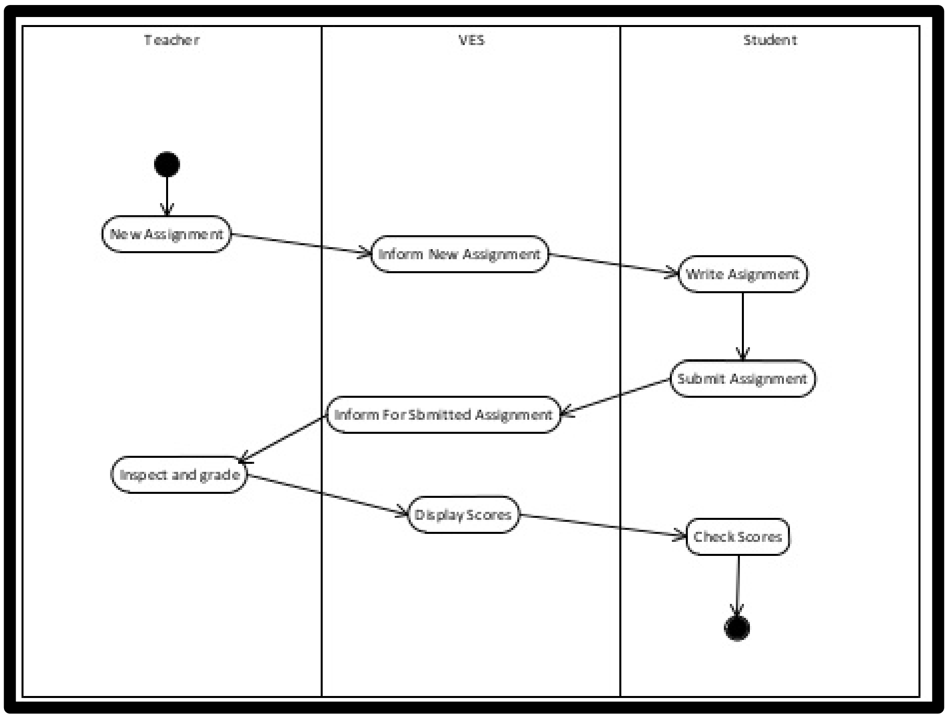


Figure 3.6 Activity Diagram for writing assignment

3.3.4 Sequence Diagram

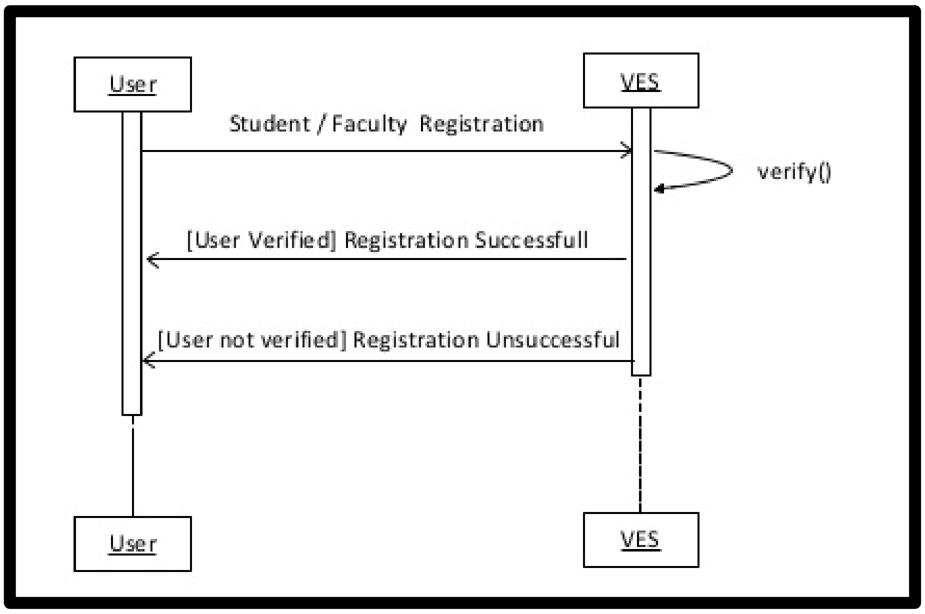

User
VES
Student / Faculty Registration
verify()
[User Verified] Registration Successfull
[User not verified] Registration Unsuccessful
User
VES

Figure 3.7 Sequence Diagram (Registration)

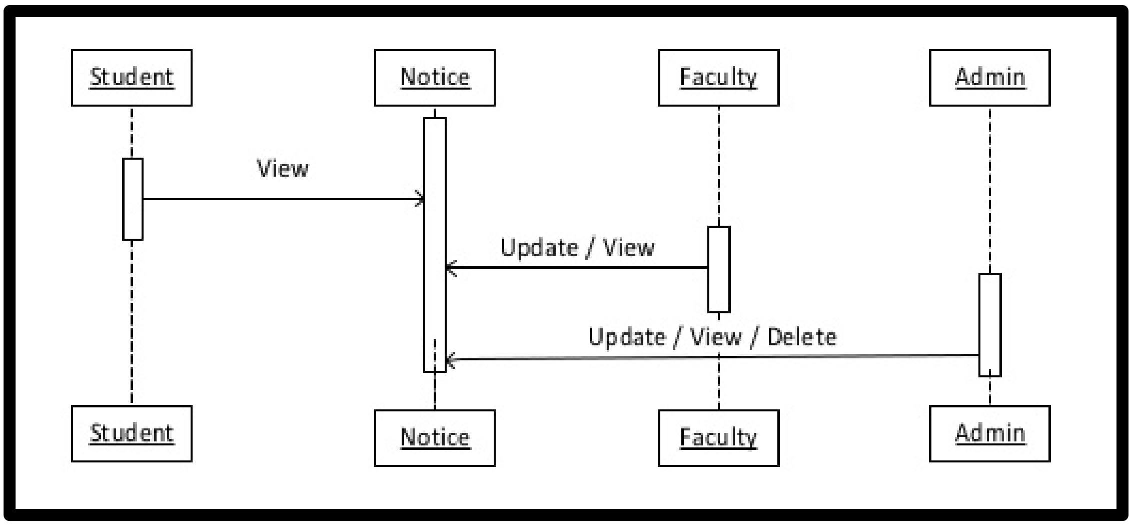


Figure 3.8 Sequence Diagram (Notice)

3.4 Screenshots

3.4.1 Splash Screen: Splash screen displayed on starting app. The screen is animated to bring a sense of organic movement when opening the app.

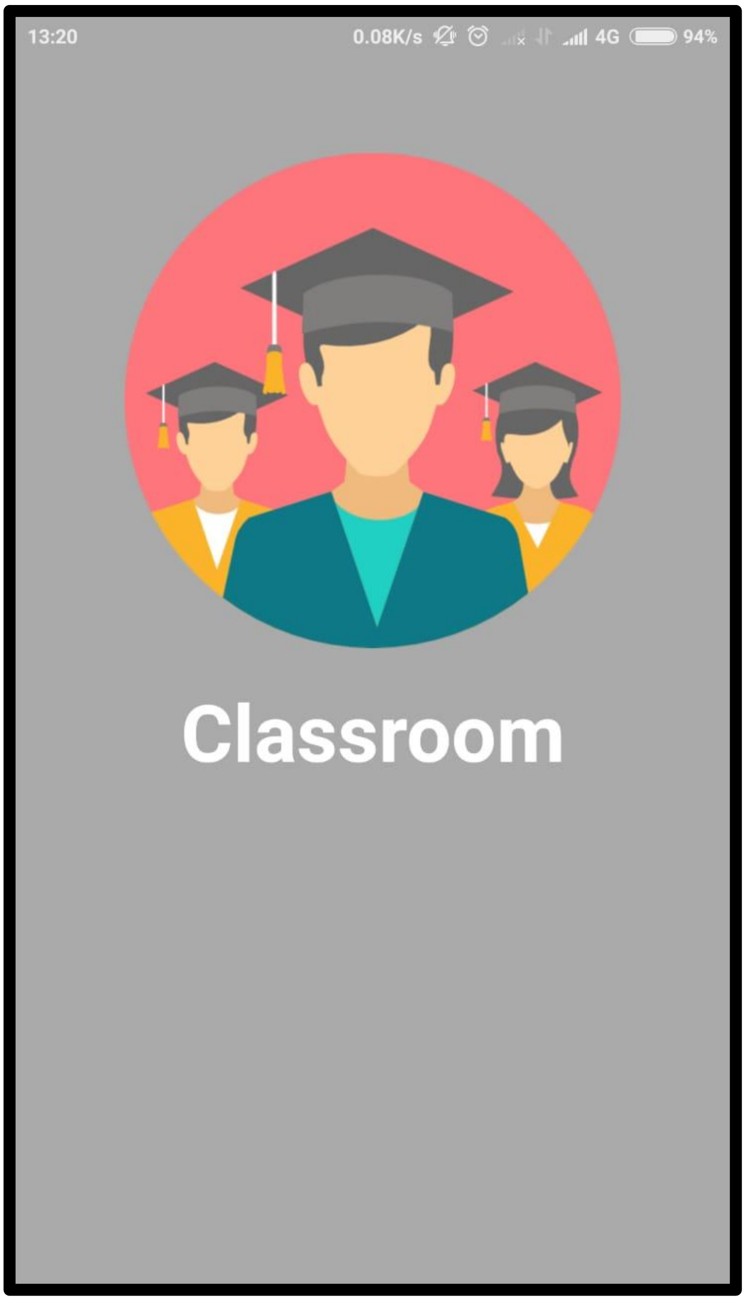


Figure 3.11 App Splash Screen

3.4.2 Home Screen:

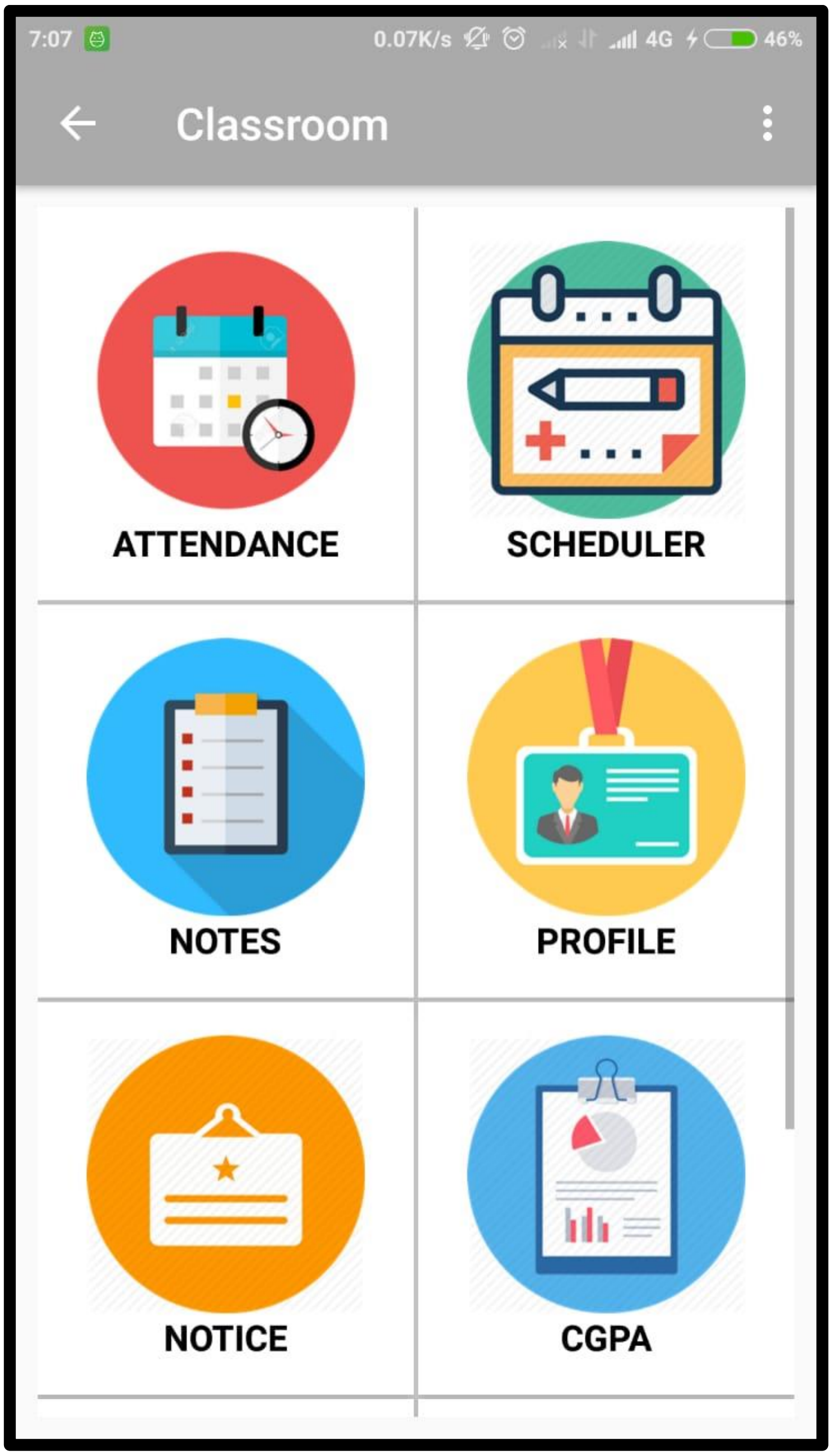


Figure 3.12 Main Screen

3.4.5 Signup Screen

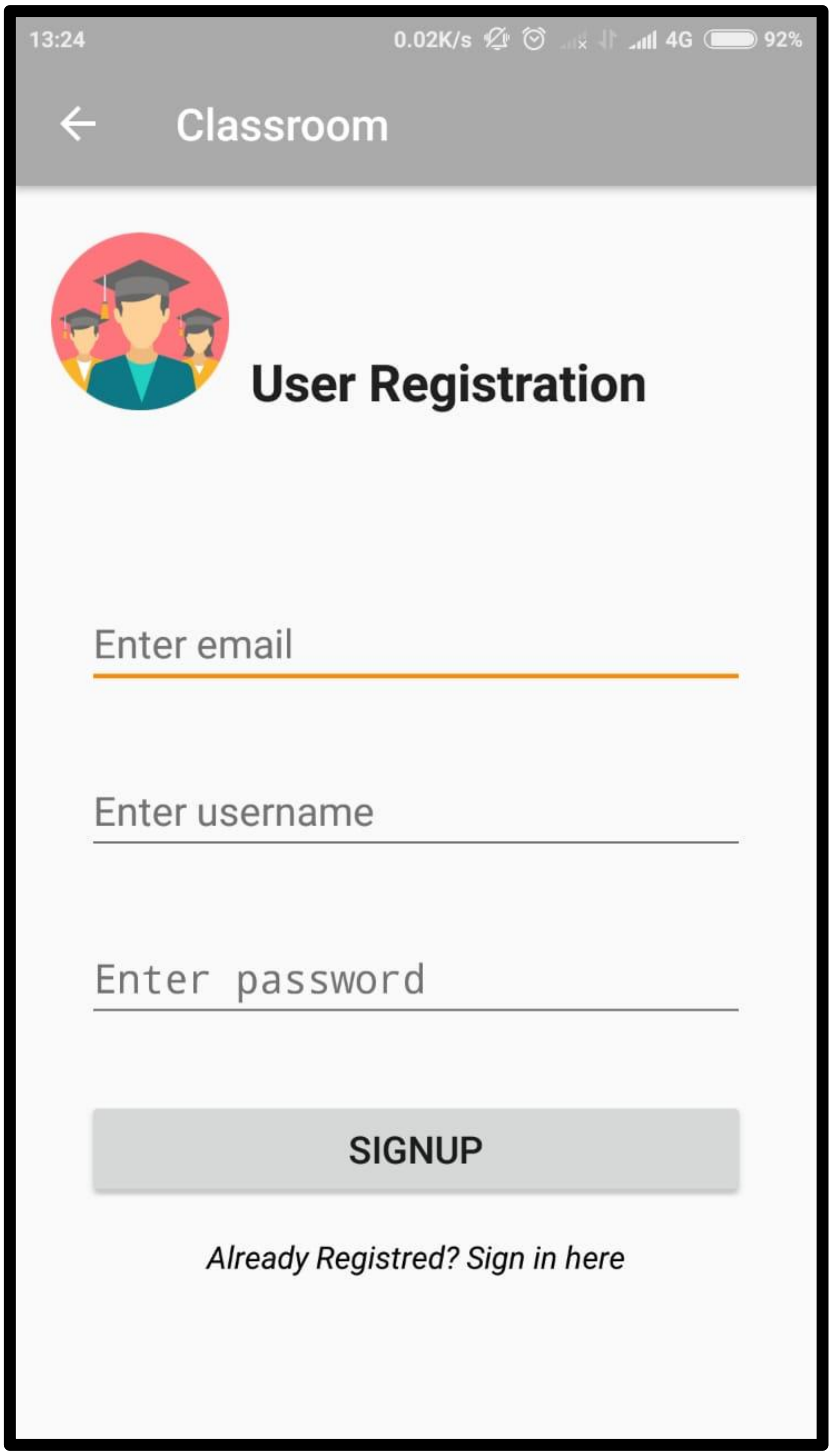


Figure 3.13 Registration Screen

3.4.3 Attendance Screen

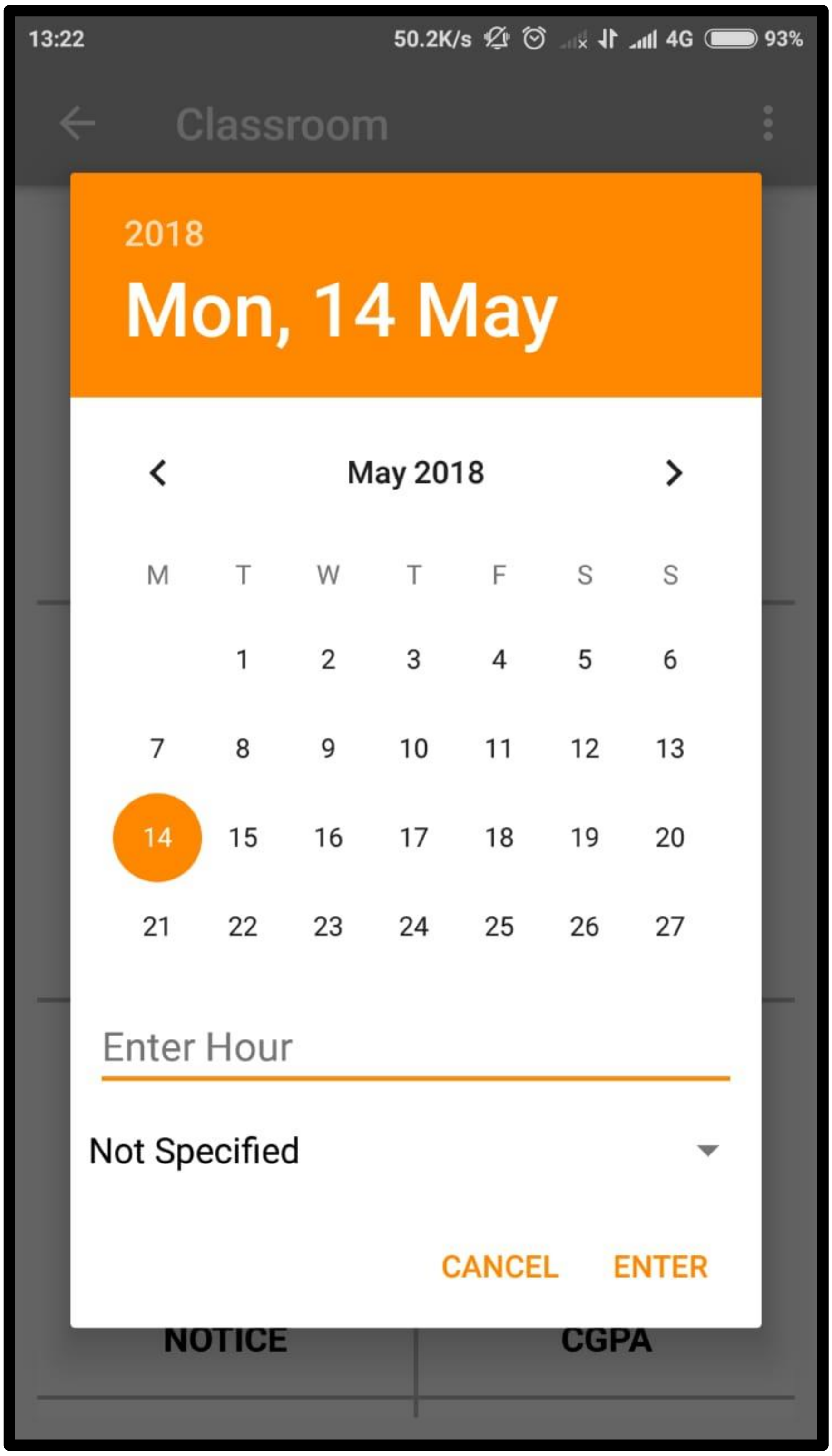


Figure 3.14 Attendance Screen

3.4.4 Notes Uploading

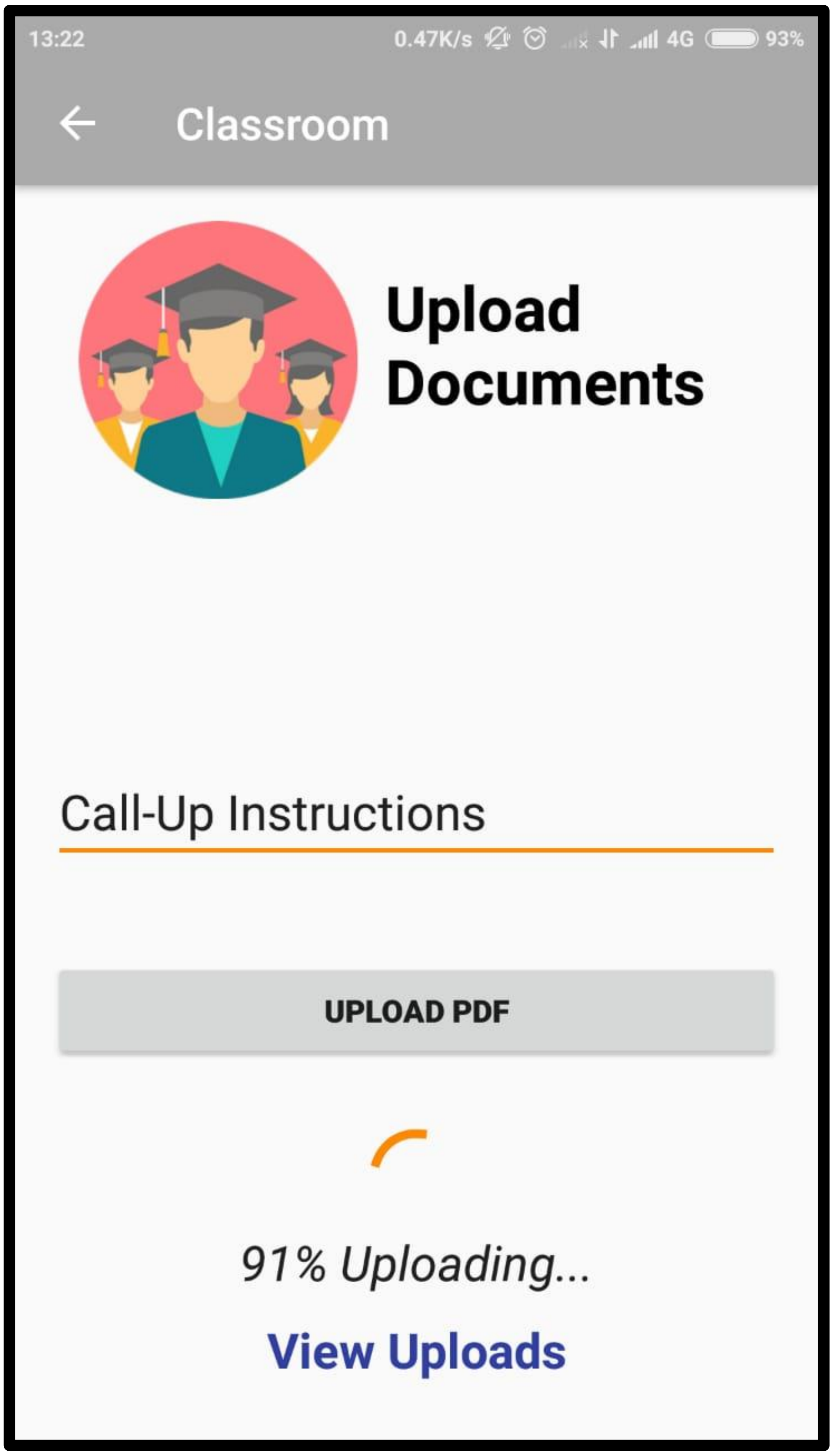


Figure 3.15 Notes Uploading Screen

CHAPTER 4
PERFORMANCE ANALYSIS

4.1 Fundamentals of Testing

The information given below document guides through important concepts that relate to Android app testing. Made by Google, an outlining of the testing tools and APIs is provided.

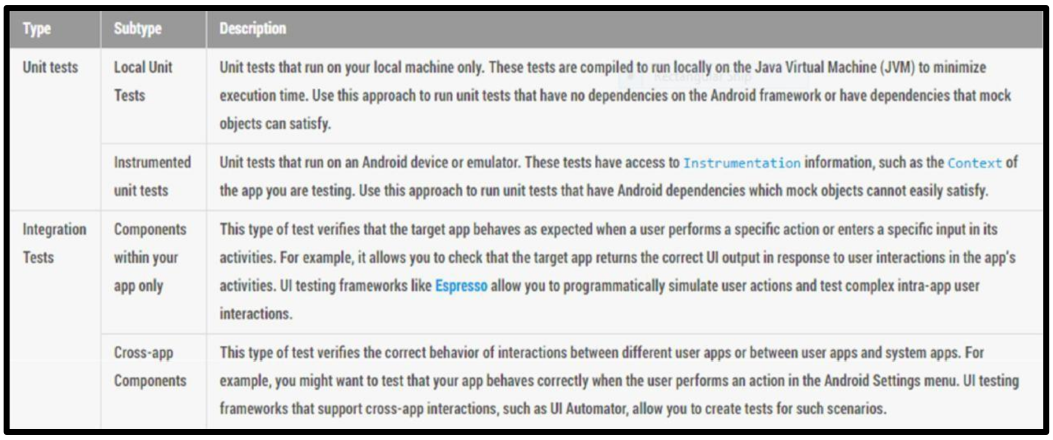
Android presents a coordinated framework encourageing to test every part of the application. Android Testing Support Library incorporates APIs for the setup and execution of test apps using an emulator.

4.1.1 Testing Concepts

JUnit is used for Android testing. JUnit testing, in essence, is a method that tests apps partly using its methods. Test methods are classified into what are known as test cases. Test suits contain group test cases.

In JUnit, a single or multiple test classes may be built. To execute them on a local machine, the test runner is used. Using Android Studio, multiple test source files can be built into an test Android app and be used to test applications on physical Android devices or the emulator.

The test code structure and the way tests are built and run in “Android Studio” are dependent on the testing being performed. Table 4.1 below showcases common Android testing types:



Type	Subtype	Description
Unit tests	Local Unit Tests	Unit tests that run on your local machine only. These tests are compiled to run locally on the Java Virtual Machine (JVM) to minimize execution time. Use this approach to run unit tests that have no dependencies on the Android framework or have dependencies that mock objects can satisfy.
	Instrumented unit tests	Unit tests that run on an Android device or emulator. These tests have access to `Instrumentation` information, such as the `Context` of the app you are testing. Use this approach to run unit tests that have Android dependencies which mock objects cannot easily satisfy.
Integration Tests	Components within your app only	This type of test verifies that the target app behaves as expected when a user performs a specific action or enters a specific input in its activities. For example, it allows you to check that the target app returns the correct UI output in response to user interactions in the app's activities. UI testing frameworks like Espresso allow you to programmatically simulate user actions and test complex intra-app user interactions.
	Cross-app Components	This type of test verifies the correct behavior of interactions between different user apps or between user apps and system apps. For example, you might want to test that your app behaves correctly when the user performs an action in the Android Settings menu. UI testing frameworks that support cross-app interactions, such as UI Automator, allow you to create tests for such scenarios.

Table 4.1 Testing Types

4.1.2 Instrumentation

Instrumentation is “a set of control methods, or hooks, in the Android system. They are responsible for controlling Android components autonomously of their regular lifecycles (the control methods).” These hooks are also in charge of controlling how apps are loaded by Android.

Figure 4.1 below demonstrates the framework for testing:

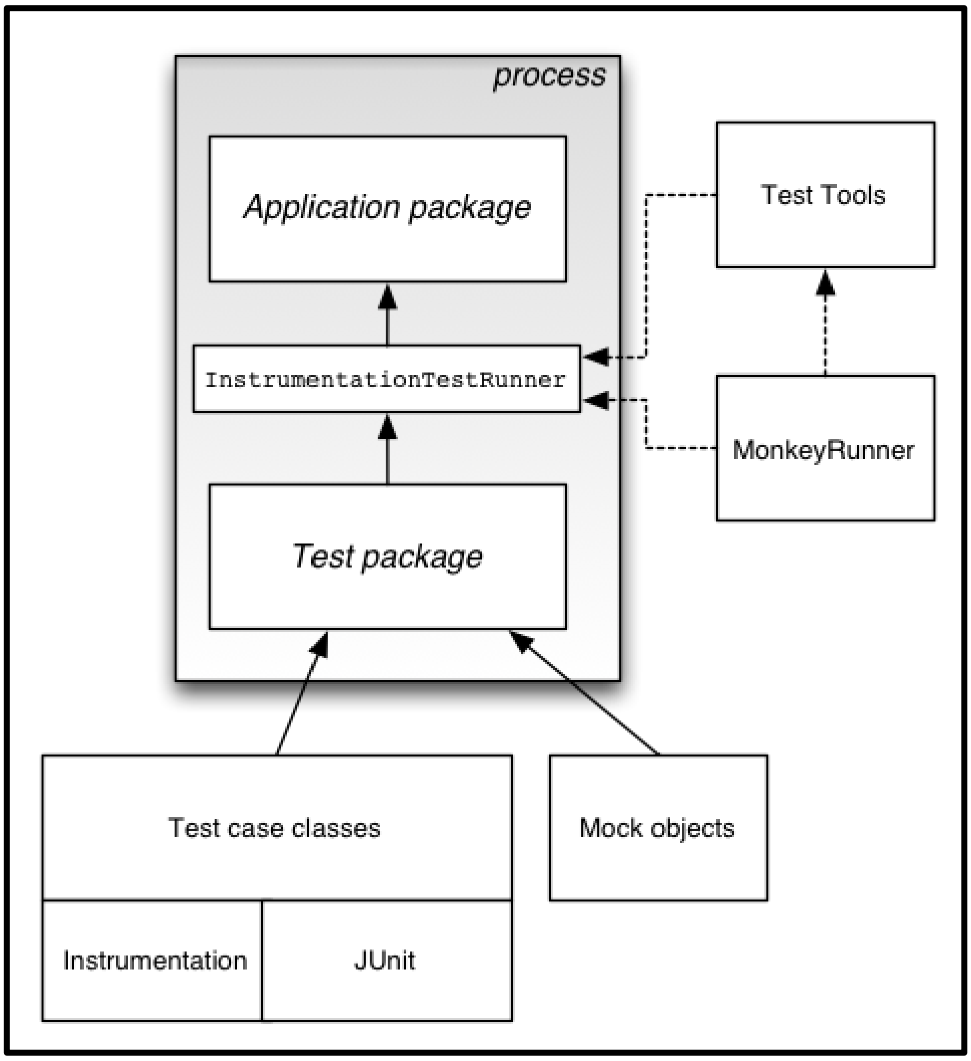
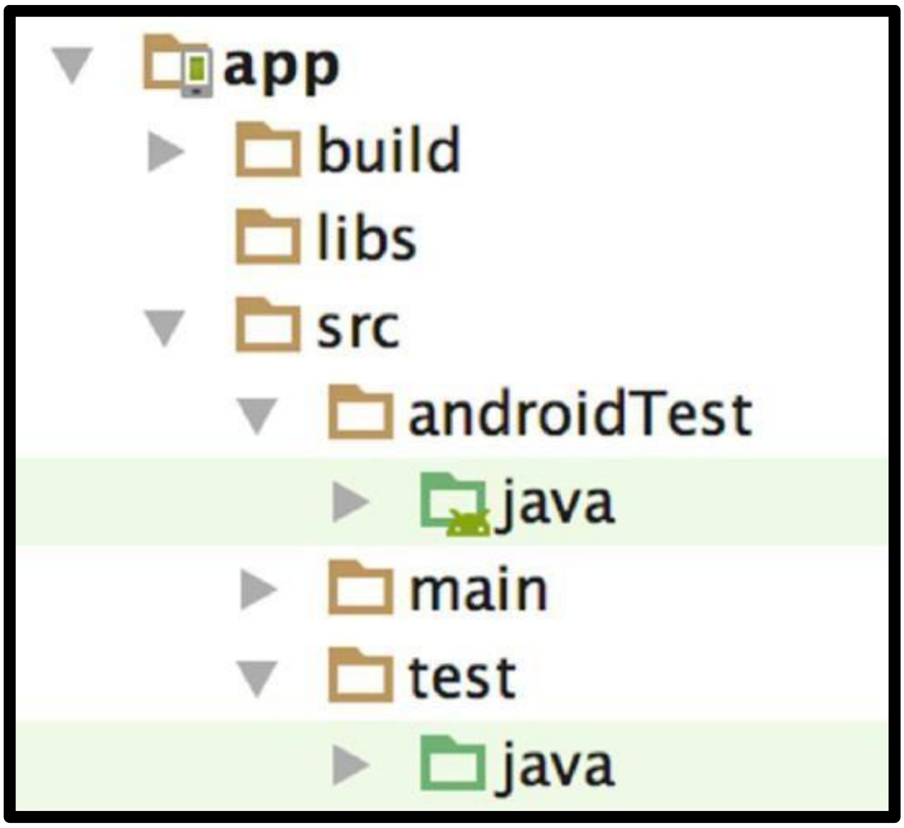


Figure 4.1 Testing Framework

Ordinarily, Android components run in system determined lifecycles. E.g., the lifecycle of an Activity object's begins when an Activity is activated by an Intent. The framework calls the object's onCreate() strategy, then the onResume() technique. At the point when the client begins another app, the framework calls the onPause() technique.



```
app
    build
    libs
    src
        androidTest
            java
        main
        test
            java
```

Figure 4.2 Default app module test directories

androidTest/ source set are common to every build variant. However, you can create additional source set directories for tests that are specific to certain build variants:

```
      src/
         main/
   androidTest/


   flavor1/
          androidTestFlavor1/


          flavor2/
```

androidTestFlavor2/

For example, when a test APK is being built for the "flavor1" version of the app, Gradle uses the androidTestFlavor1/ and androidTest/ source sets. Every tests runs the debug build. This could be changed to a different build type by making use of the testBuildType property in the module-level build.gradle file.

```
android {
    ...
    testBuildType "staging"
}
```

Gradle automatically generates manifest files for your androidTest/ source sets. Optionally, you can create your own manifest.

4.2 Testing APIs:

4.2.1 JUnit

According to Android's developer support document, "A unit or joining test class is ought to be composed as a JUnit 4 test class. JUnit is the most mainstream and broadly utilized unit testing framework for Java. The framework offers a supportive strategy to perform standard setup, teardown, and announcement exercises in your test."

A fundamental JUnit 4 test class is a Java class that contains at least one test strategies. A test strategy starts with the @Test explanation. Containing the practice and confirm a solitary usefulness (that is, a consistent unit) in the segment that need to be tested.

The accompanying bit demonstrates a case “JUnit 4 mix” test using the “Espresso APIs” to play out tick activity on a UI component, at that point verifies whether a normal string is shown.

```
@RunWith(AndroidJUnit4.class)
@LargeTest
public class MainActivityInstrumentationTest {
        @Rule
        public ActivityTestRule mActivityRule = new
        ActivityTestRule<>( MainActivity.class);
        @Test
        public void sayHello(){
                onView(withText("Say hello!")).perform(click());
                onView(withId(R.id.textView)).check(matches(withText("Hello, World!")));
```

4.2.2 Android Testing Support Library APIs

A collection of APIs is provided by the Android Testing Support Library. These APIs enable us to rapidly build and run test codes. These include, but are not limited to, JUnit 4 and functional UI tests. The below instrumentation-based APIs are provided by the library that prove beneficial when tests are to be automated:

AndroidJUnitRunner

A test runner that is JUnit 4-compatible for Android.

Espresso

A UI testing framework; suitable for functional UI testing within an app.

UI Automator

A UI testing framework suitable for cross-app functional UI testing between both system and installed apps.

CHAPTER 5

CONCLUSIONS

5.1 Conclusions

We learnt how the proposed system is better and overall more efficient than the existing system. It is highly compatible and greatly reduces human effort. Through the proposed system, the standard will maintain security and also provide features that are not included in the existing system. This system provides a rather simple way to maintain records. It provides an easy way for interaction between students and college faculties. Students can also improve interaction skills by making use of the proposed system.

The proposed system will also help the college to manage their records and save natural resources. This will also get the job done in less time compared to existing systems. All the users receive the information without delays because of real time data accessing and updating. It helps in reaching every student and faculty in the college instantaneously. The data which is stored on the database will help the management take major decisions, keeping in mind the suggestions and ideas taken from the system.

The proposed system will also help the students to get their queries solved by the answers of other users such as fellow students and respected faculties of the college. Since there will be many users on system we planned to make it secure and spam free using various algorithms.

5.2 Future Scope

In the future this system can be implemented to automate most of the educational system and it can be designed for cross platform use. This would greatly improve the level of penetration and transparency that the several facilities and services of education have, and that are being provided to both the students and faculty over-the-air.

The application will significantly help in simplifying and speeding up the management and communication process. It offers good level of safety and a structure that lessens manual work and reduces usage of resources the conventional procedure demands. The proposed system tenders a better way of computing and displaying actions with receptive and striking user-interfaces.

Hence, with the help of several literature surveys and through examination of the prevailing systems, we have made the conclusion that using the proposed application will not only assist in the automating the academy learning process, but similarly helping to digitize the system and therefore helping in the efficient deployment of resources.

REFERENCES

[1] Rakhi Joshi, V. V. Shete, S. B. Somani, “Android Based Smart Learning and Attendance Management System”, International Journal of Advanced Research in Computer and Communication Engineering (IJARCCE), Vol. 4 Issue 6, June 2015

[2] A.J.Kadam, Aradhana Singh, Komal Jagtap, Srujana Tankala, “Mobile Web Based Android Application for College Management System”, International Journal of Engineering and Computer Science (IJECS) ISSN: 2319-7242, Volume 6 Issue 2, Feb. 2017, Page No. 20206-20209

[3] Manasi Kawathekar, Kirti K. Bhate, Pankaj Belgoankar, “An Android Application for Student Information System”, International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) Volume 4 Issue 9, September 2015

[4] Prof. Sagar Rajebhosale, Mr. Shashank Choudhari, Mr. Sachin Patil, Mr. Akshay Vyavahare, Mr. Sanket Khabiya, “SMART CAMPUS – An Academic Web Portal with Android Application”, International Research Journal of Engineering and Technology (IRJET), Volume 3 Issue 4, Apr-2016

[5] Pallavi Mohadikar, Nasrin Mulani, Afnan Shaikh, Rachna Sable, “College Parent Interaction using Android Application”, International Journal of Computer Science and Network (IJCSN), Volume 4 Issue 1, February 2015