# ASSIGNMENT NO. A5

## Problem Statement:

A Mobile App for Calculator having Trigonometry functionality is to be designed and tested. The data storage uses 1.text files, 2. XML Use latest open source software modeling, Designing and testing tool/ Scrum-it. Implement the design using HTML-5/Scala/ Python/Java/C++/Rubi on Rails. Perform Positive and Negative testing.

## Learning Objectives:

1. To study how to create a Mobile App.
2. To study how to store data using text file or XML file.

## Learning Outcomes:

Learnt about android application and scientific calculator.

## Software and Hardware Requirements:

1. 64-bit operating System(Linux)
2. Android SDK
3. Dalvik Virtual machine

# Theory

## Mobile App:

A mobile app is a computer program designed to run on mobile devices such as smartphones and tablet computers. Most such devices are sold with several apps bundled as pre-installed software, such as a web browser, email client, calendar, mapping program, and an app for buying music or other media or more apps. Some pre-installed apps can be removed by an ordinary uninstall process, thus leaving more storage space for desired ones. Where the software does not allow this, some devices can be rooted to eliminate the undesired apps.

Mobile native apps stand in contrast to software applications that run on desktop computers, and to web applications which run in mobile web browsers rather than directly on the mobile device.

## Development:

Developing apps for mobile devices requires considering the constraints and features of these devices. Mobile devices run on battery and have less powerful processors than personal computers and also have more features such as location detection and cameras. Developers also have to consider a wide array of screen sizes, hardware specifications and configurations because of intense competition in mobile software and changes within each of the platforms.

Mobile application development requires use of specialized integrated development environments. Mobile apps are first tested within the development environment using emulators and later subjected to field testing. Emulators provide an inexpensive way to test applications on mobile phones to which developers may not have physical access.

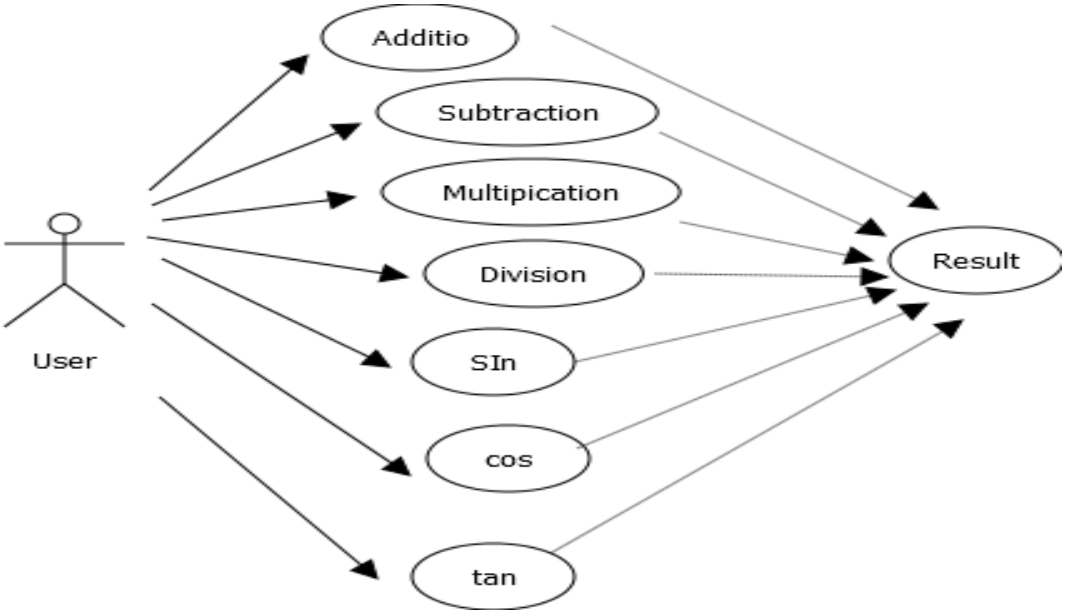
Mobile user interface (UI) Design is also essential. Mobile UI considers constraints and contexts, screen, input and mobility as outlines for design. The user is often the focus of interaction with their device, and the interface entails components of both hardware and software. User input allows for the users to manipulate a system, and device's output allows the system to indicate the effects of the users' manipulation. Mobile UI design constraints include limited attention and form factors, such as a mobile device's screen size for a user's hand. Mobile UI contexts signal cues from user activity, such as location and scheduling that can be shown from user interactions within a mobile application. Overall, mobile UI design's goal is primarily for an understandable, user-friendly interface.

Mobile UIs, or front-ends, rely on mobile back-ends to support access to enterprise systems. The mobile back-end facilitates data routing, security, authentication, authorization, working off-line, and service orchestration. This functionality is supported by a mix of middleware components including mobile app servers, Mobile Backend as a service (MBaaS), and SOA infrastructure.

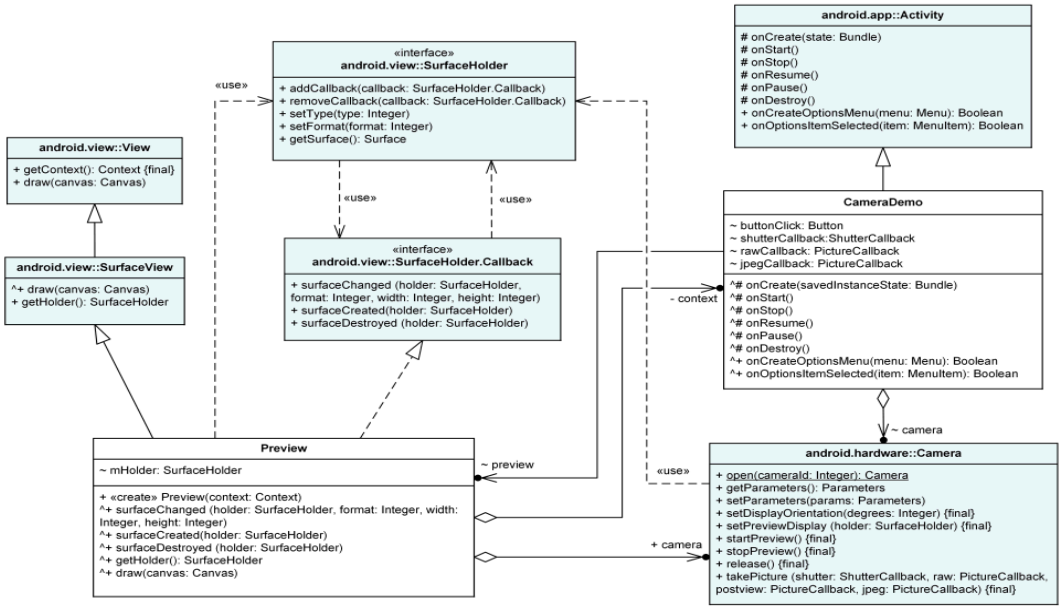
## Mathematical Model

```
S = {s, e, I, o, f, DD, NDD, success, failure}
s = {Initial state of system}
  = start android sdk
e = end of system
I = {input of system}
I = {0-9,f}
O = {Output of system}i.e calculator with various functionality
DD = {Deterministic Data }
   = input numbers
NDD = {Non-Deterministic Data }

f = {sin(),cos(),tan(),clearall(),view calculations()}
sin()={Calculate sine of the number}
cos()={Calculate cosine of the number}
tan()={Calculate tangent of the number}
clearall()={Clear all the calculations}
view calculations()={View calculations till date}

success = Correct output is displayed.
failure = Incorrect output is displayed.
```

**Use case Diagram**



## Class Diagram




## Positive Testing

Positive testing is a testing technique to show that a product or application under test does what it is supposed to do. Positive testing verifies how the application behaves for the positive set of data.

1. Enter numbers for calculation
2. Provide scientific operations i.e sin, cos and tan.

## Black Box Testing

Black-box testing is a method of software testing that examines the functionality of an application without peering into its internal structures or workings. This method of test can be applied to virtually every level of software testing: unit, integration, system and acceptance. PRECONDITIONS:

1. Scientific operations

## White Box testing

White-box testing is a method of testing software that tests internal structures or workings of an application, as opposed to its functionality (i.e. black-box testing). White box testing is a testing technique, that examines the program structure and derives test data from the program logic/code. The other names

of glass box testing are clear box testing, open box testing, logic driven testing or path driven testing or structural testing.

1. Input by user i.e numbers

## Conclusion

We have successfully created a mobile app for calculator having trigonometry functionality.