**CHAPTER – 1:**

# INTRODUCTION

**1.1 Overview Problem Statement**

Lack of immediate retrievals: In the conventional system, information is distributed across several files. This might also lead to data redundancy with repetition of the same information in various files. In the event of a complex or nested query, the search has to scan several files, thus making procurement of requested query results very cumbersome.

Maintenance of Accuracy and Reliability issues: With redundancy comes consistency issues as the update of information in a single record should be echoed in all records containing the same information. Also, atomicity issues i.e. completion of a transaction in totality or nothing at all; has to be maintained. This is difficult in a multi-file system.

Lack of prompt update: Updates associated with a record in a file is to be reflected in all records wherein the particular record is present. This concurrent update poses the problem of time lag. Errors in commit operation to some particular files cause the grave issue of data inconsistency. Error prone manual calculation: Manual calculations are error prone and relatively immensely time consuming, in spite of which they may result in generation of incorrect information. Verification is another overhead, which can be saved through efficient design and implementation.

**1.2 Project Introduction**

In modern society, where the population is increasing the people are dependent on their own transport system such as cars, bikes etc. So, the people usually purchase the vehicles on monthly instalments that is EMI. So, to calculate the EMI that to loan EMI an inbuilt calculator should be there so that it is easy for the customers to calculate the EMI.

The loan calculator helps to calculate the equated monthly instalments (EMIs) that have to paid by the lender every month till the loan is fully paid. The EMI is based on the loan amount, the tenure, and interest rate. On the given loan amount, interest rate and for a specific duration, the calculator will let us know how much EMI should to be paid.

Various banks and third-party websites offer a Loan EMI Calculator that allows us to calculate the Equated Monthly Instalments that must be paid. We can make use of the Loan

EMI Calculator to estimate the amount we have to pay each month towards the car loan.

The online EMI Calculator will calculate the monthly instalments as well as provide us a detailed loan repayment. Basic details such as the repayment tenure, principal amount and the rate of interest must be entered to calculate the EMI.

#### 1.3 Objectives

The objective of this application is to propose development of android that:

1. Make it with a simple feature and run smoothly.
2. To produce a system with increased efficiency.
3. To create an easy-to-use application.

**1.4 Android OS**

Android is a mobile operating system (OS) currently developed by Google, based on the Linux kernel and designed primarily for touchscreen mobile devices such as smartphones and tablets. Android's user interface is mainly based on direct manipulation, using touch gestures that loosely correspond to real-world actions, such as swiping, tapping and pinching, to manipulate on-screen objects, along with a virtual keyboard for text input. In addition to

Touch screen devices, Google has further developed Android TV for televisions, Android Auto for cars and Android Wear for wrist watches, each with a specialized user interface. Variants of Android are also used on notebooks, game consoles, digital cameras, and other electronics. Initially developed by Android, Inc., which Google bought in 2005, Android was unveiled in 2007, along with the founding of the Open Handset Alliance – a consortium of hardware, software, and telecommunication companies devoted to advancing open standards for mobile devices. As of July 2013, the Google Play store has had over one million Android applications ("apps") published, and over 50 billion applications downloaded. An April–May 2013 survey of mobile application developers found that 71% of developers create applications for Android, and a 2015 survey found that 40% of full-time professional developers see Android as their priority target platform.

## 1.4.1 Android Architecture

We studied the Android system architecture. Android system is a Linux-based system, Use of the software stack architecture design patterns. As shown in Figure 1, the Android

architecture consists of four layers: Linux kernel, Libraries and Android runtime, Application framework and Applications [5-8]. Each layer of the lower encapsulation, while providing call interface to the upper.

## Applications:

Android app will be shipped with a set of core applications including client, SMS program, calendar, maps, browser, contacts, and others. All these application programs are developed in Java.

## Application Framework:

The developer is allowed to access all the API framework of the core programs. The application framework simplifies the reuse of its components. Any other app can release its functional components and all other apps can access and use this component (but have to follow the security of the framework). Same as the users can be able to substitute the program components with this reuse mechanism

## Libraries and Android Runtime

The library is divided in to two components: Android Runtime and Android Library. Android Runtime is consisted of a Java Core Library and Dalvik virtual machine. The Core Library provides Java core library with most functions. Dalvik virtual machine is register virtual machine and makes some specific improvements for mobile device. Android system library is supporting the application framework; it is also an important link connecting between application framework and Linux Kernel. This system library is developed in C or C++ language. These libraries can also be utilized by the different components in the Android system. They provide service for the developers through the application framework.

## Linux Kernel

The kernel system service provided by Android inner nuclear layer is based on Linux 2.6 kernel; Operations like internal storage, process management, internet protocol, bottom drive and other core service are all based on Linux kernel.

## 1.4.2 Android Studio

Android Studio is exclusively designed for 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

**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.

**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.

**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).

**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 add shell to get permissions.

**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.

**AVD Manager:** It is used to create virtual devices of any desired API level to support higher level SDK’s incasing 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 – 2:**

**SYSTEM REQUIREMENT SPECIFICATION**

## Minimum Hardware Requirements:

 64-bit Microsoft® Windows® 8/10.

 X86\_64 CPU architecture; 2nd generation Intel Core or newer, or AMD CPU with support for a Windows Hypervisor.

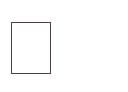
 S8 GB RAM or more.

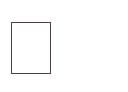
 8 GB of available disk space minimum (IDE + Android SDK + Android Emulator)

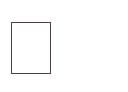
* + - 1280 x 800 minimum screen resolutions.

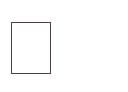
## Minimum Software Requirements

### Windows requirements:

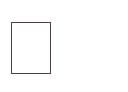
 Microsoft Windows 7/8/10 (32-bit or 64-bit)

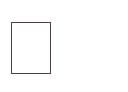
 3 GB RAM minimum, 8 GB RAM recommended (plus 1 GB for the Android Emulator)

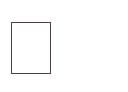
 2 GB of available disk space minimum, 4 GB recommended (500 MB for IDE plus

1.5 GB for Android SDK and emulator system image)  1280 x 800 minimum screen resolution.

### Mac OS requirements:

 Mac OS X 10.10 (Yosemite) or higher, up to 10.13 (High Sierra)

 3 GB RAM minimum, 8 GB RAM recommended (plus 1 GB for the Android Emulator)

 2 GB of available disk space minimum, 4 GB recommended (500 MB for IDE plus

1.5 GB for Android SDK and emulator system image)

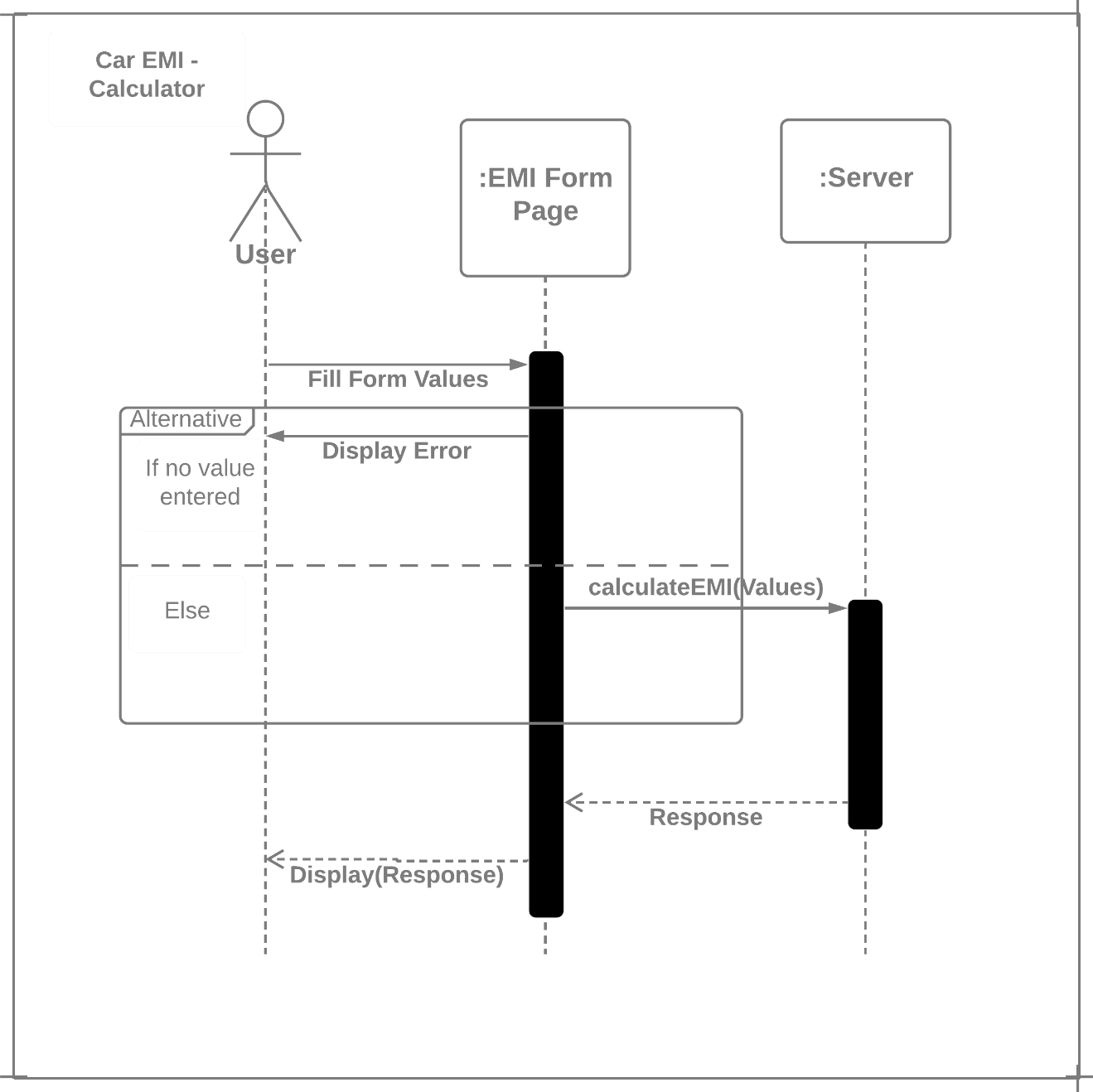
* + - 1280 x 800 minimum screen resolutions.

## CHAPTER - 3:

# DESIGN

#### 3.1 Sequence Diagram

A sequence diagram is a type of interaction diagram because it describes how and in what order a group of objects works together. Sequence diagrams are sometimes known as event diagrams or event scenarios.



**LOAN**

**EMI**

Figure 3.1: Sequence diagram of EMI Loan Calculator

#### 3.2 Activity Diagram

Activity diagram is another important diagram in UML to describe the dynamic aspects of the system. Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. The control flow is drawn from one operation to another.

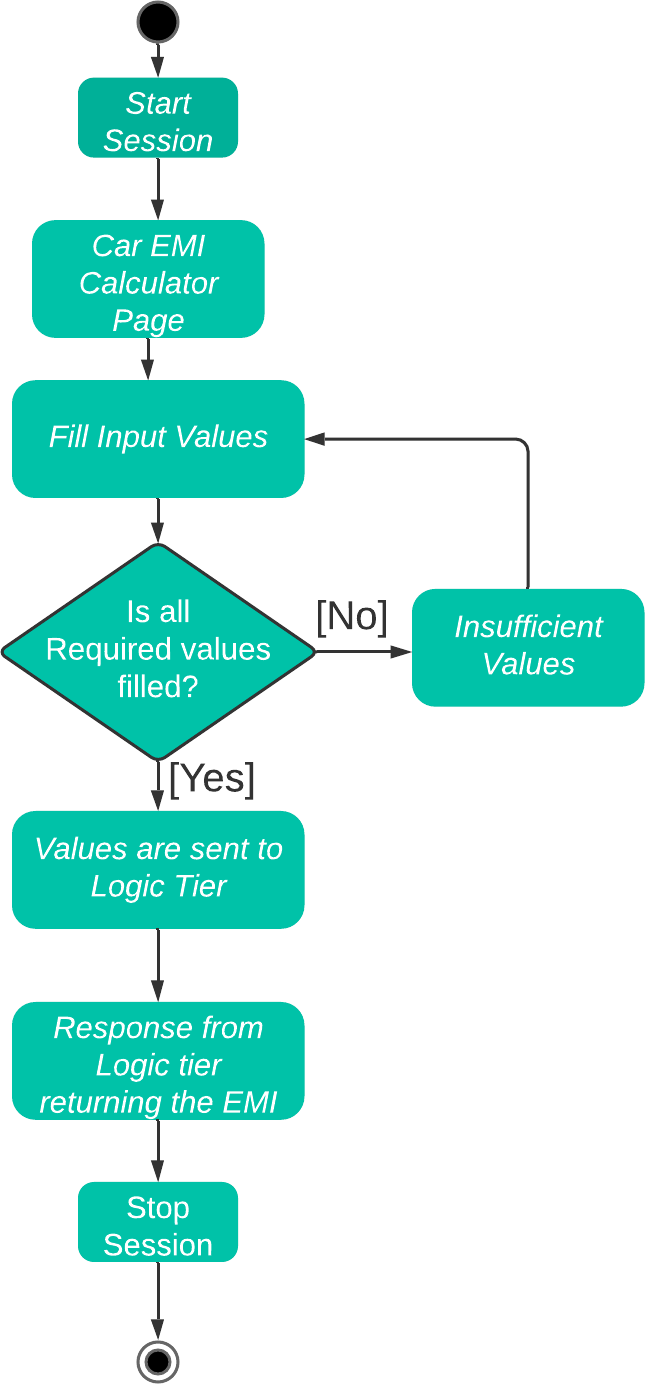


Figure 3.2: Activity Diagram of EMI Calculator

**Chapter - 4:**

**IMPLEMENTATION**

**4. How to Play**

**JAVA CODE:**

**MainActivity.java**

package com.example.emi;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;

import android.text.TextUtils;

import android.view.View;

import android.widget.Button;

import android.widget.EditText;

public class MainActivity extends AppCompatActivity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

final EditText P=(EditText)findViewById(R.id.editText);

final EditText R=(EditText)findViewById(com.example.emi.R.id.editText2);

final EditText N=(EditText)findViewById(com.example.emi.R.id.editText3);

final EditText emi=(EditText)findViewById(com.example.emi.R.id.editText5);

Button emibtn=(Button)findViewById(com.example.emi.R.id.button);

emibtn.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

String loan=P.getText().toString();

String rate=R.getText().toString();

String tenure=R.getText().toString();

if(TextUtils.isEmpty(loan))

{

P.setError("Enter loan amount");

return;

}

if(TextUtils.isEmpty(rate))

{

R.setError("Enter rate");

return;

}

if(TextUtils.isEmpty(tenure))

{

N.setError("Enter the loan tenure");

return;

}

float p=Float.parseFloat(loan);

float r=Float.parseFloat(rate)/100;

int t=Integer.parseInt(tenure);

double e=p\*r\*(Math.pow((1+r),t))/(Math.pow((1+r),t)-1);

emi.setText(String.valueOf(e));

}

});

Button reset=(Button)findViewById(com.example.emi.R.id.button2);

reset.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

P.setText("");

R.setText("");

N.setText("");

}

});

}

}

**Chapter - 5**

### TESTING

#### 5.1 Introduction to Testing

The reason behind testing is to find errors. Every program or software has errors in it, against the common view that there are no errors in it if the program or software is working.

Executing the programs with intention of finding the errors in it is therefore testing; hence a successful test is one which finds errors. Testing is an activity; however, it is restricted to being performed after the development after the development phase is complete, but is carried parallel with all stages of the system development, starting with requirement specification.

## 5.2 Testing Plan

**5.2.1 Requirement Traceability:**

As the most interested portion is whether the system is meeting its requirements or not, for that testing should be planned so that all requirements are individually tested. We have to check out that output of certain combinations of inputs gives the desirable results or not. The requirements specification gives the path to get the desirable result.

**5.2.2 Testing Schedule:**

We have tested each procedure back-to-back so that errors and omissions can be found as early as possible.

## 5.3 Testing Strategy:

A strategy for the software testing integrates software test case design methods into a well-planned series of steps that result in the successful construction of software. The strategy provides a road map that describes the steps to be conducted as part of testing. When these steps are planned and then undertaken, very much efforts, time and resources are required. A software testing strategy should be flexible enough to promote a customized testing approach. At that same time, it must be rigid enough to promote reasonable planning and management tracking as the project progresses.

A software testing strategy has following characteristics:

* Testing begins at the component level and works outward towards the integration of the entire computer-based system.
* Different testing techniques are appropriate at different points in time. Testing and Debugging are different activities but debugging must be accommodated in any testing strategy. We checked entire project thoroughly so not even a single mistake would be there.

## 5.4 Test Cases

|  |  |  |
| --- | --- | --- |
| **INPUT** | **EXPECTED OUTPUT** | **ACTUAL OUTPUT** |
| Enter Valid values in the  EMI form | The values should be entered | Pass |
| If any of the form field is empty | Toast displaying enter values of that particular field | Pass |
| Enter invalid values in the form | Throws an exception | Pass |

**CHAPTER – 6:**

**SNAPSHOTS**

**CHAPTER – 7:**

### CONCLUSION AND FUTURE ENHANCEMENT

## 7.1 Conclusion

* The EMI calculator gives an opportunity to calculate the monthly instalments that should be repaid after the disbursal of car loan. It helps to assess which scheme matches the customer affordability.
* The EMI calculator also provides a comprehensive view of the loan repayment. With the help of iconographic such as a pie chart or a table, the EMI calculator represents the proportion of the amounts of interest rate, principal and the processing fee, if any of the total repayment value.
* The calculator also allows us to view the various segments of the repayment value, individually and helps in determining the car loan eligibility. It presents a break of the total payable amount into the loan amount; total interest amounts payable and the processing fee.

## 7.2 Future Enhancement

EMI or the Equated Monthly Instalment, at the basic level, refers to the fixed amount that you would be entitled to repay monthly to the loan provider until the loan matures. EMI is a part of every loan, in this case the loan, is computed by principal amount and a rate of interest. In this scenario, loan EMI is the most important element in loan repayment since it is the EMI that a borrower would be liable to pay, for a short - to medium- term, depending on the choice of tenure. A lower EMI gives more lucrative schemes to a borrower. But at the same time, a higher EMI for a shorter tenure might be more sensible. What is evident by now is that the preference for EMI is subjective. Whatever the case may be the evaluation of monthly loan EMI using the loan EMI calculator is indispensable.

**BIBLIOGRAPHY**

**Books:**

1. Erik Hellman, “Android Programming – Pushing the Limits”, 1st Edition Wiley India Pvt. Ltd, 2014.
2. Dawn Griffiths and David Griffiths, “Head First Android Development”, 1st

Edition, 12 O’Reilly SPD Publishers, 2015.

1. Google Developer Training, "Android Developer Fundamentals Course – Concept

Reference”, Google Developer Training Team, 2017.

**Links:**

1. [https://www.gitbook.com/book/google-developer-training/android-developer-](https://www.gitbook.com/book/google-developer-training/android-developer-fundamentals-course-concepts/details) fundamentals-course-concepts/details



1. [https://www.w3schools.com](https://www.w3schools.com/)
2. [https://stackoverflow.com](https://stackoverflow.com/)