A

Project Report

on

Doodh Dairy App

Submitted to

Sant Gadge Baba Amravati University, Amravati

Submitted in partial fulfillment of

the requirements for the Degree of

Bachelor of Engineering in

Electronics and Telecommunication Engineering

Submitted by

Shruti Prashant Jumale	**Srushti Moreshwar Chohatkar**
(PRN: 193120153)	(PRN: 193120436)
Sejal Gopal Ganesh	**Disha Sanjay Nagpal**
(PRN: 193120270)	(PRN: 193120292)

Under the Guidance of

Dr. B. P. Harne

Professor, E & TC Dept.

Department of Electronics & Telecommunication Engg.

Shri Sant Gajanan Maharaj College of Engineering,

Shegaon – 444 203 (M.S.)

2022-2023

Department of Electronics & Telecommunication Engineering
Shri Sant Gajanan Maharaj College of Engineering,
Shegaon – 444203, Maharashtra, India
(Recognized by AICTE, Accredited by N.B.A, New Delhi)

Certificate

This is to certify that the project report entitled **"Doodh Dairy App"** is hereby approved as a creditable study carried out and presented by

Shruti P. Jumale	(PRN: 193120153)
Srushti M. Chohatkar	(PRN: 193120436)
Sejal G. Ganesh	(PRN: 193120270)
Disha S. Nagpal	(PRN: 193120292)

in a manner satisfactory to warrant of its acceptance as a pre-requisite in a partial fulfillment of the requirements for the degree of Bachelor of Engineering in Electronics& Telecommunication Engineering of Sant Gadge Baba Amravati University, Amravati during the Session 2022-23.

[illegible]
Dr. B. P. Harne
Project Guide

[illegible]
Prof. K. S. Vyas
Internal Examiner

[illegible] 31/05/23
External Examiner

[illegible]
Dr. M. N. Tibdewal
Professor & Head, E & TC Dept.

[illegible]
Dr. S. B. Somani
Principal

Abstract

Dairies are the most critical part of management used in societies and their surroundings. Milk plays a major role on the daily basis in our lives and offers a good source of nutrition like calcium, especially for children as well as older people. This research focused on the online Android application ‘Doodh Dairy’ for the sake of the Dairy owners and their customers. The development of this application ‘Doodh Dairy’ was aimed at addressing the problem which was, the dairy owner has to follow the traditional method to store dairy data of how much dairy products are sold daily to the customers. In COVID-19 we have to follow the rules of social distancing. In dairy, it is impossible to maintain social distancing as Customers have to be physically present at the dairy and due to this effect we have designed and developed an online dairy application. By doing a survey we came to know about the problem of Sellers. It is difficult for them to maintain data in hard copies and need to store a soft copy of data, So the system was thus developed to provide the solution to these problems and help its users to manage their work in a fast and friendly manner. Its feature includes, allowing sellers to update, record delete and save data. Our finding highlights the importance of the Doodh Dairy Application. We conclude with a discussion of the theory's implication for study and suggestions for future research.

Keywords: Android Application, Android Studio.

.

Acknowledgement

We would like to take this opportunity to express our heartfelt thanks to our guide **Dr. B. P. Harne** for her esteemed guidance and encouragement, especially through difficult times. His suggestions broaden our vision and guided us to succeed in this work. We are also very grateful for his guidance and comments while designing part of our project and learnt many things under his leadership. Also we would like to thank to Prof. (Dr.) M. N. Tibdewal, Head of Electronics and Telecommunication Department, all teaching and non-teaching staff of EXTC Department for their encouragement and suggestions for our project.

We extend our thanks to Dr. S. B. Somani, Principal, Shri Sant Gajanan Maharaj, Collegeof Engineering, for his valuable support.

We sincerely thank to all our friends, who helped us directly or indirectly in completing our project work. We would like to express our appreciation for the wonderful experiencewhile completions of this project work.

Place: SSGMCE, Shegaon

Shruti P. Jumale

Srushti M. Chohatkar

Sejal.G.Ganesh

Disha S. Nagpal

Abbreviations

API -Android Application Interface

XML -Extensible Markup Language

SDK -Software Development Kit

JDK -Java Development Kit

JVMS -Java Virtual Machine Specification

JLS -Java Language Specification

SE - Standard Edition

List of Figures

List of Tables

Contents

Chapter 1

Introduction

The use of smartphones has significantly increased over that of laptop or desktop computers. Nowadays, customers want their applications to be mobile-friendly in every industry, including the financial services, medical care, and insurance sectors. The process by which application software is created for compact handheld devices like smartphones, tablets, etc. is known as mobile application development. Android currently holds the largest market share among mobile applications. The Android ecosystem as a whole got numerous advancements.

The IDE known as Android Studio was created exclusively for the building of Android applications. Numerous businesses are spending a lot of capital developing Android applications. As compared to other OSes, the Android operating system has undergone an incredible amount of updates, leading to a web-based service that is significantly distinct from the mobile operating system's first release. The standard IDE for creating Android applications is now

Android Studio.

We achieved our project objective by calculating daily sales of dairy products. In this application, our system offers a mobile app that allows the user to place an order by simply following a few simple steps. From the dairy seller's list, the buyer can choose the dairy products they need. Customers can access the most recent prices for dairy products through our 'Doodh Dairy' app. The price of the products may be added, removed, or changed by the seller. The seller can also monitor the buyer's daily activities. The 'Doodh Dairy' application offers both online and offline payment methods. Here, the application offers balance sheets that display all payment information.

A diary's owner should be aware of the significant and careful effort required to maintain it because diaries involve a lot of paperwork. If their owners don't regularly check them and update their diary entries daily, diaries become useless. However, the ‘Doodh Dairy Application’ enables the automation of administrative procedures, reduces the use of paper documents, maximizes work time, and suggests a general turning point in terms of task management and compliance efficiency.

1.1 Motivation

In COVID-19, it was difficult for the dairy sellers to manage their work as each and every person has to follow social distancing. Apart from this, by taking the reviews from the dairy sellers we came to know, they need an android application which will replace their traditional working methods.

1.2 Literature Survey

For the management of milk distributing activities, there are numerous applications already available. Every application has unique traits, drawbacks, and benefits. These applications (apps) were created by taking the requirements of a specific dairy owners.
[1] Field Survey provides us the knowledge of the requirements of the customers.
Ghandi et al. [2] presented the different methods, standards, and guidelines used in the "Mobile application development - a practical approach," which was also listed.
Xhafa et al. [3] defined and examined mobile learning techniques from a technological and educational point of view Li et al. [4] presented the layout of the client terminal.
Chou et al. [5] stated the necessity and essentiality of mobile technology in our daily lives.
Field Survey
Name-________________
Dairy Name- _____________
Phone number- ____________
Sr.no Questions Yes No

1. Do you face any difficulty while maintaining the orders?
2. Are you satisfied with this process for managing your dairy?
3. Are your costumers satisfied with this process?
4. Are you using existing online dairy apps available on Play Store?
5. Do you feel of having an user friendly system for running your bussiness more smoothly?

1.3Objectives

The objective of this project is to develop an adaptive, sustainable, platform for the dairy sellers which will make their work easier.

a) To manage activities related to dairy work.
b) To reduce manual work in the dairy field.
c) To facilitate easily maintenance for the dairy owners.
d) To prevent and reduce human error.

1.4 System Overview

This section discusses what are the requirements of this project, system architecture and the required technologies to create it. It define a view's position in ConstraintLayout, you add at least one horizontal and one vertical constraint for the view. Each constraint represents a connection or alignment to another view, the parent layout, or an invisible guideline. Each constraint defines the view's position along the vertical or horizontal axis. Each view must have a minimum of one constraint for each axis, but often more are necessary.

When you drop a view into the Layout Editor, it stays where you leave it even if it has no constraints. This is only to make editing easier. If a view has no constraints when you run your layout on a device, it is drawn at position [0,0] (the top-left corner).

In figure 1, the layout looks good in the editor, but there's no vertical constraint on view C. When this layout draws on a device, view C horizontally aligns with the left and right edges of view A, but it appears at the top of the screen because it has no vertical constraint.

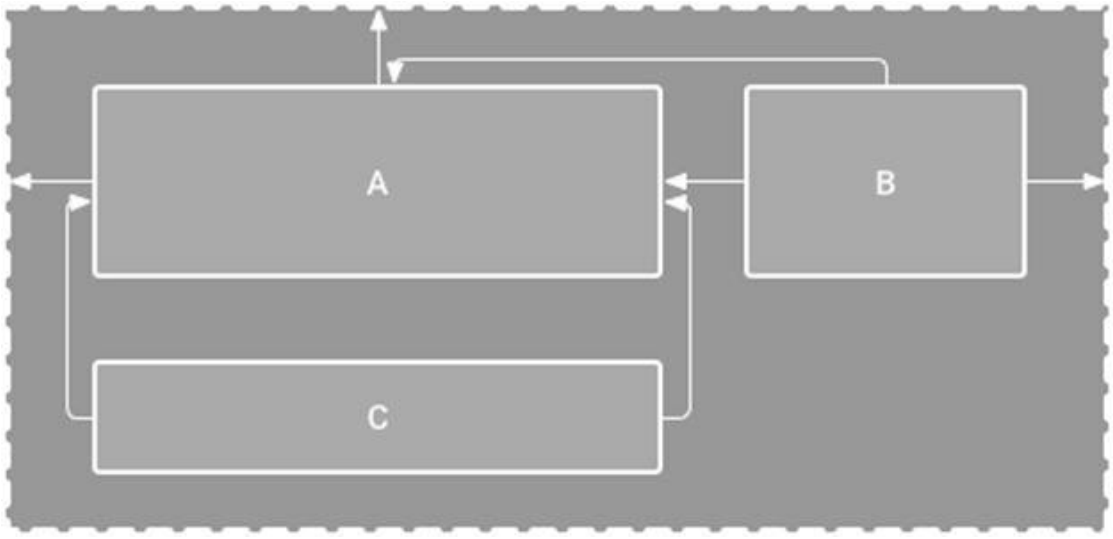


Figure 1. The editor shows view C below A, but it has no vertical constraint.

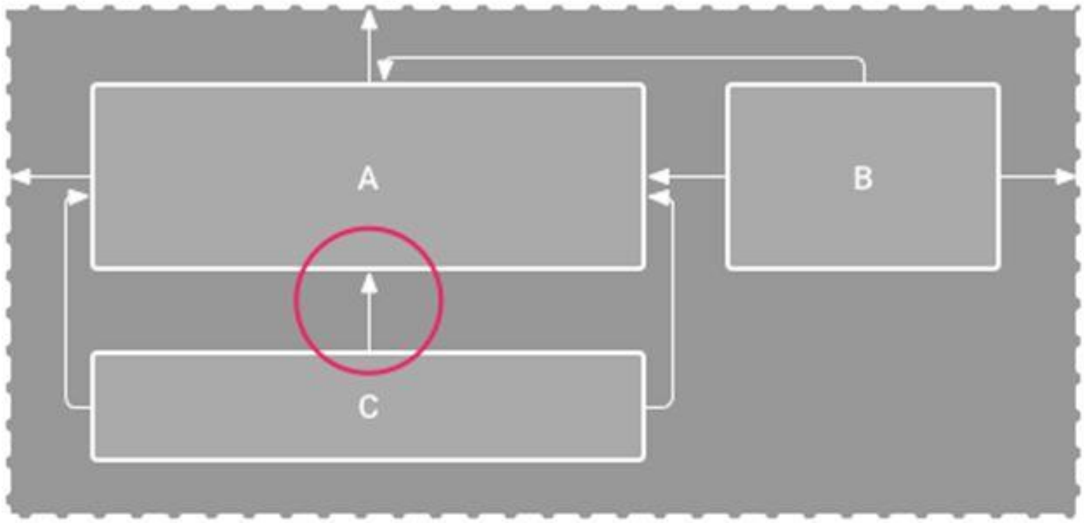


Figure 2. View C is now vertically constrained below view A.

Although a missing constraint doesn't cause a compilation error, the Layout Editor indicates missing constraints as an error in the toolbar. To view the errors and other warnings, click **Show Warnings and Errors.** To help you avoid missing constraints, the Layout Editor automatically adds constraints for you with the Autoconnect and infer constraints features.

Add ConstraintLayout to your project

To use ConstraintLayout in your project, proceed as follows:

1. Ensure you have the maven.google.com repository declared in your settings.gradle file:

GroovyKotlin

```
dependencyResolutionManagement {
...
repositories {
google()
}
)
```

2. Add the library as a dependency in the module-level build.gradle file, as shown in the following example. The latest version might be different than what is shown in the example.

GroovyKotlin

```
dependencies {
implementation "androidx.constraintlayout:constraintlayout:2.2.0-alpha09"
// To use constraintlayout in compose
implementation "androidx.constraintlayout:constraintlayout-compose:1.1.0-alpha09"
}
```

3. In the toolbar or sync notification, click **Sync Project with Gradle Files**.

Now you're ready to build your layout with ConstraintLayout.

Convert a layout

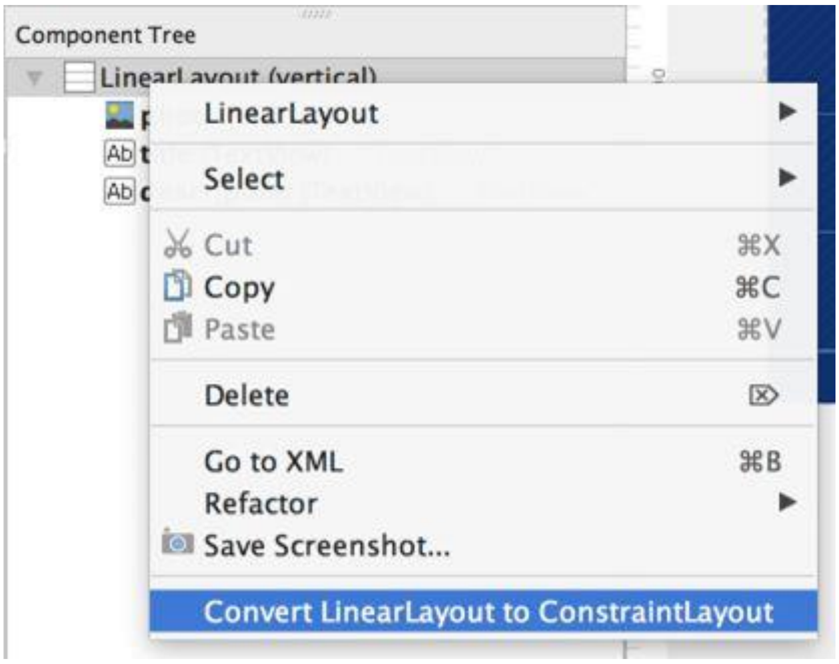


Figure 3. The menu to convert a layout to ConstraintLayout.

To convert an existing layout to a constraint layout, follow these steps:

1. Open your layout in Android Studio and click the **Design** tab at the bottom of the editor window.
2. In the Component Tree window, right-click the layout and click **Convert LinearLayout** to **ConstraintLayout.**

Create a new layout

To start a new constraint layout file, follow these steps:

1. In the Project window, click the module folder and select **File > New > XML > Layout XML**.
2. Enter a name for the layout file and enter "androidx.constraintlayout.widget.ConstraintLayout" for the **Root Tag.**
3. Click Finish.

1.5 Project Outline

The dairy application that is the subject of this paper is called Doodh Dairy App. Java and XML are the languages used for the application's development on the 8.0 platform.

On the backend, SQLite is used.

The first step is to download Android Studio 3.0, which includes the Android SDK and Android Virtual Device. The studio has been installed, and its environment has been set up, including the API level. The process for setting the API level is to start a new project by clicking on File>New>New Project. Enter the values listed below in the Create by selecting the "New Project" window. Name of the programme: "Doodh Dairy App".

The package name will be created using all of these data: package com. mini project.dairy_milk_application. This contains all of the information related to the development of Android; it is referred to as the "Android package file," which must be uploaded to the Google Play store.

Enter the location where you want the application file saved last.

Developers must select Empty Activity as their project template if they want to create a new application with different requirements. Select Next. Accept the Main

Activity (Default Activity) name. To finish, click. All of these are allegedly. Android package files that you must upload to the Google Play store.

The path where you want to save will be entered in the final field. You must choose the blank activity from the activity module if you want to customise your application. You will add more items that you want to display to your application, similar to this Activity. 'activity_main.xml' and 'content_main.xml' are the first two files you'll see when you open your application in Android Studio. Both essentially accomplish the same task, but the activity_main.xml file contains the fundamental layout from when you chose Basic Activity. You can edit the content in content_main.xml.

Widgets are a group of options in Android where you can drag and drop objects. A rendering of the layout as it will appear on the screen rather than XML code is displayed in Android Studio. You must first open MainActivy.java in order to define the behaviour of your application. These tabs are located in the Application>Java section. Run your application once you've finished setting up your code or layout design. On an Android virtual device known as an emulator, you should first test your application. The next step is to test your application on a real device after a successful test. In order to proceed, you must first connect your device via a USB cable to your development computer. Enable USB debugging on your device by opening the Developer option. You can then select Run from the toolbar in Android Studio's toolbar after clicking the application module in the project window. You must choose your device in the Select Deployment Target window before clicking the OK button. On the connected device, Android Studio launches and installs the application. The application you created is currently running on your device, as you can see.

DESIGN MODULES OF THE DOODH DAIRY APPLICATION

The viewer side consists of a splash screen, registration screen, and buyer dashboard which includes a View profile, Buy product, and Balance amount. The categories' specifics are detailed below.

SPLASH SCREEN

The very first splash screen that appears when an Android application is launched blinks for 15 seconds.

Application names are displayed on the splash screen alongside the logo.

Basically, the Android Studio tool has an inbuilt feature for the splash screen, and after selecting the module, it opens the inbuilt design of the screen, which can be modified as needed.

REGISTRATION

Later on, this window will redirect us either to a Buyer login or to a Seller login. Here the system requires a database which can be used for storing or retrieving data, processing transactions, or various machine learning calculations. In our application, we have used four database tables, they are the Seller table, buyer table, product table, and purchase table. From this buyer and seller login page, the user can also register themselves. While registering, the system will store the information of the user in the buyer table or in the seller table (depending upon whether the user is the buyer or the seller). If the user has already registered themselves they can enter the user id and password in their specific domain, the system will check whether the entered data is valid or not (by reading the data from the buyer table or the seller table).

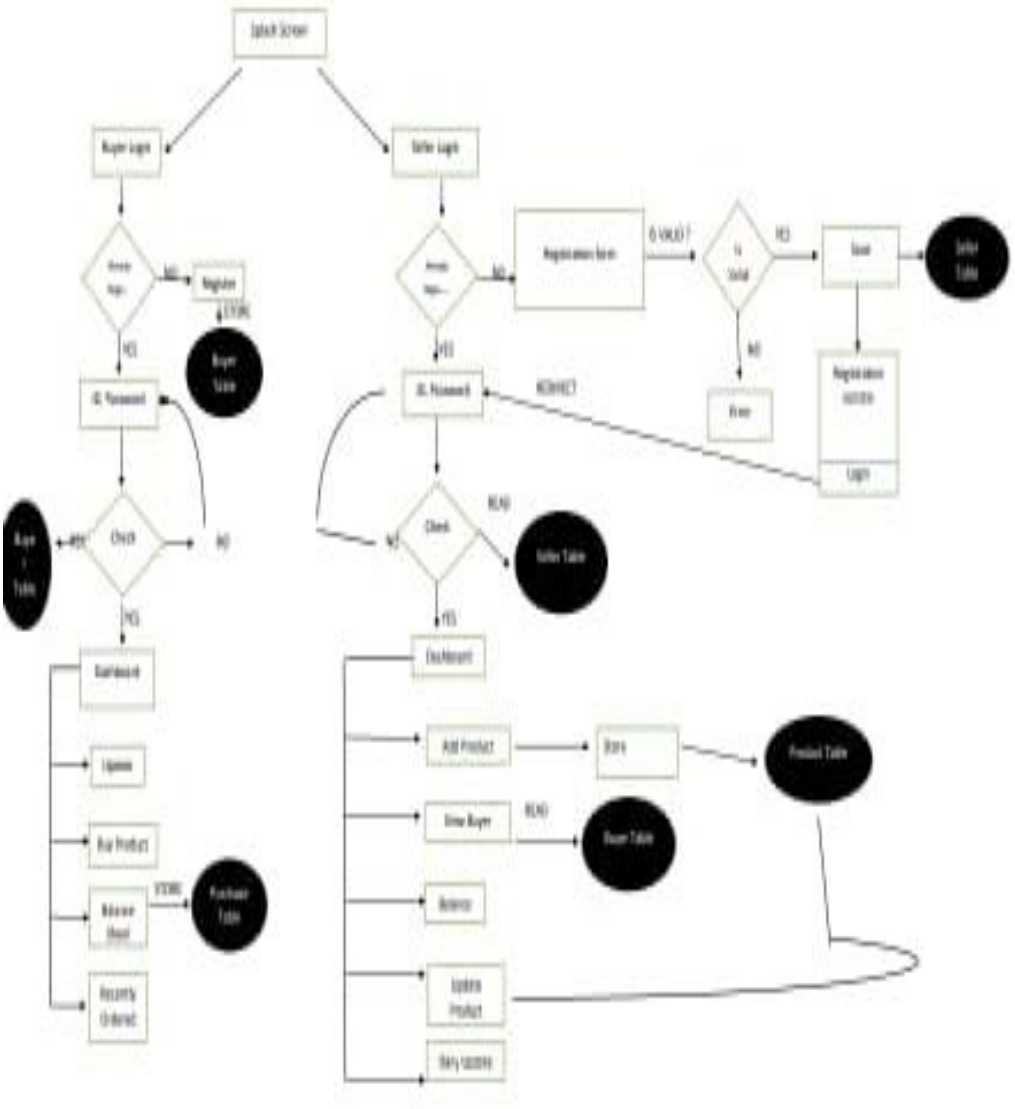
HOME SCREEN AND MENU SCREEN

Moving further, if the user is here to buy dairy products then the system will redirect them to the dashboard which contains: View Profile, buy the product, balance amount and one important feature called 'recent purchase' which will show the recently ordered products so it will be convenient for the user to order the product again. In the 'view profile' section he/she can also update their information. In the Buy product, the user can start adding the product and the system will display the total amount. The 'Balance amount' screen will show the payment-related details which will be stored in the purchase table.

SELLER SIDE

The seller dashboard will display components named add product, view buyer, daily update, view product, Balance sheet, and orders. The seller can add the available dairy products in the add product section (Doodh, paneer Dahi, etc) which will be stored in the product table. View buyer and view product screens will read the data from the buyer table and product table respectively. If the end user i.e. seller wants to make some changes (altering the prices) in the already added product then he/she can click on daily update. The balance sheet is present in the system so that the seller will get daily, monthly and yearly updates about her/his sales. The order screen will display the requirements of the buyer which he/she has added in the Buy product section.

Flowchart



Splash Screen
Buyer Login
Seller Login
NO
Register
STORE
Buyer Table
YES
ID, Password
Check
NO
YES
Buyer Table
Dashboard
Update
Buy Product
STORE
Purchase Table
Recently Ordered
Registration form
IS VALID ?
Is Valid
YES
Seller Table
NO
Error
Registration success
Login
CORRECT
READ
Check
Seller Table
YES
Dashboard
Add Product
Store
Product Table
View Buyer
READ
Buyer Table
Balance
Update Product

Chapter 2

Methodology

This paper is based on the available survey data. To understand the trends and patterns in the Indian dairy market, a thorough analysis of the literature was conducted. Survey data were investigated to find and explain different factors that affect a mobile application's acceptance in the context of Indian milk marketing, newer trends, and the future market. Sales reports that were made available from the survey reports served as the data sources. It is a case study investigation. A "5" question questionnaire with predetermined interview questions was used for the milk seller. It is an honest attempt to shed light on this special type of dairy business. According to the survey, the vast majority of sample respondents are illiterate and adhere to traditional values. To select samples, the researcher took into account factors such as geographic location, age, buffalo population, experience level, turnover, etc. It is a case the study, so it has its own limitations regarding research methodology is a concern.

Chapter 3

Andriod Application Interface (API)

An **application programming interface** (**API**) is a way for two or more computer programs to communicate with each other. It is a type of software interface, offering a service to other pieces of software.[1] A document or standard that describes how to build or use such a connection or interface is called an *API specification*. A computer system that meets this standard is said to *implement* or *expose* an API. The term API may refer either to the specification or to the implementation.

In contrast to a user interface, which connects a computer to a person, an application programming interface connects computers or pieces of software to each other. It is not intended to be used directly by a person (the end user) other than a computer programmer who is incorporating it into the software. An API is often made up of different parts which act as tools or services that are available to the programmer. A program or a programmer that uses one of these parts is said to *call* that portion of the API. The calls that make up the API are also known as subroutines, methods, requests, or endpoints. An API specification *defines* these calls, meaning that it explains how to use or implement them.

One purpose of APIs is to hide the internal details of how a system works, exposing only those parts a programmer will find useful and keeping them consistent even if the internal details later change. An API may be custom-built for a particular pair of systems, or it may be a shared standard allowing interoperability among many systems.

There are APIs for programming languages, software libraries, computer operating systems, and computer hardware. APIs originated in the 1940s, though the term did not emerge until the 1960s and 1970s. Contemporary usage of the term API often refers to web APIs,[2] which allow communication between computers that are joined by the internet. Recent developments in APIs have led to the rise in popularity of microservices, which are loosely coupled services accessed through public APIs

3.1 Libraries and frameworks

The interface to a software library is one type of API. The API describes and prescribes the "expected behavior" (a specification) while the library is an "actual implementation" of this set of rules.

A single API can have multiple implementations (or none, being abstract) in the form of different libraries that share the same programming interface.

The separation of the API from its implementation can allow programs written in one language to use a library written in another. For example, because Scala and Java compile to compatible bytecode, Scala developers can take advantage of any Java API.[15]

API use an vary depending on the type of programming language involved. An API for a procedural language such as Lua could consist primarily of basic routines to execute code, manipulate data or handle errors while an API for an object-oriented language, such as Java, would provide a specification of classes and its class methods.[16][17] Hyrum's law [18] states that "With a sufficient number of users of an API, it does not matter what you promise in the contract: all observable behaviors of your system will be depended on by somebody." Meanwhile, several studies show that most applications that use an API tend to use a small part of the API.[19]

Language bindings are also APIs. By mapping the features and capabilities of one language to an interface implemented in another language, a language binding allows a library or service written in one language to be used when developing in another language.[20]

Tools such as SWIG and F2PY, a Fortran-to-Python interface generator, facilitate the creation of such interfaces.[21]

An API can also be related to a software framework: a framework can be based on several libraries implementing several APIs, but unlike the normal use of an API, the access to the behavior built into the framework is mediated by extending its content with new classes plugged into the framework itself.

Moreover, the overall program flow of control can be out of the control of the caller and in the framework's hands by inversion of control or a similar mechanism.[22][23]

3.2 Operating systems

An API can specify the interface between an application and the operating system.[24] POSIX, for example, provides a set of common API specifications that aim to enable an application written for a POSIX conformant operating system to be compiled for another POSIX conformant operating system.

Linux and Berkeley Software Distribution are examples of operating systems that implement the POSIX APIs.[25]

Microsoft has shown a strong commitment to a backward-compatible API, particularly within its Windows API (Win32) library, so older applications may run on newer versions of Windows using an executable-specific setting called "Compatibility Mode".[26]

An API differs from an application binary interface (ABI) in that an API is source code based while an ABI is binary based. For instance, POSIX provides APIs while the Linux Standard Base provides an ABI.[27][28]

3.3 Remote APIs

Remote APIs allow developers to manipulate remote resources through protocols, specific standards for communication that allow different technologies to work together, regardless of language or platform. For example, the Java Database Connectivity API allows developers to query many different types of databases with the same set of functions, while the Java remote method invocation API uses the Java Remote Method Protocol to allow invocation of functions that operate remotely but appear local to the developer.[29][30]

Therefore, remote APIs are useful in maintaining the object abstraction in object-oriented programming; a method call, executed locally on a proxy object, invokes the corresponding method on the remote object, using the remoting protocol, and acquires the result to be used locally as a return value.

A modification of the proxy object will also result in a corresponding modification of the remote object.[31]

3.4 Web APIs

Main article: Web API

Web APIs are a service accessed from client devices (Mobile Phones, Laptop, etc.) to a web server using the Hypertext Transfer Protocol (HTTP). Client devices send a request in the form of an HTTP request, and are met with a response message usually in JavaScript Object Notation (JSON) or Extensible Markup Language (XML) format. Developers typically use Web APIs to query a server for a specific set of data from that server.

An example might be a shipping company API that can be added to an eCommerce-focused website to facilitate ordering shipping services and automatically include current shipping rates, without the site developer having to enter the shipper's rate table into a web database. While "web API" historically has been virtually synonymous with web service, the recent trend (so-called Web 2.0) has been moving away from Simple Object Access Protocol (SOAP) based web services and service-oriented architecture (SOA) towards more direct representational state transfer (REST) style web resources and resource-oriented architecture (ROA).[32] Part of this trend is related to the Semantic Web movement toward Resource Description Framework (RDF), a concept to promote web-based ontology engineering technologies. Web APIs allow the combination of multiple APIs into new applications known as mashups.[33]

In the social media space, web APIs have allowed web communities to facilitate sharing content and data between communities and applications. In this way, content that is created in one place dynamically can be posted and updated to multiple locations on the web.[34] For example, Twitter's REST API allows developers to access core Twitter data and the Search API provides methods for developers to interact with Twitter Search and trends data.[35]

3.5 Design

The design of an API has a significant impact on its usage.[4] First of all, the design of programming interfaces represents an important part of software architecture, the organization of a complex piece of software.[36] The principle of information hiding describes the role of programming interfaces as enabling modular programming by hiding the implementation details of the modules so that users of modules need not understand the complexities inside the modules.[37] Aside from the previous underlying principle, other metrics for measuring the usability of an API may include properties such as functional efficiency, overall correctness, and learnability for novices.[38] One straightforward and commonly adopted way of designing APIs is to follow Nielsen's heuristic evaluation guidelines. The Factory method pattern is also typical in designing APIs due to their reusable nature.[39] Thus, the design of an API attempts to provide only the tools a user would expect.[4]

Synchronous versus asynchronous[edit]

An application programming interface can be synchronous or asynchronous. A synchronous API call is a design pattern where the call site is blocked while waiting for the called code to finish.[40] With a asynchronous API call, however, the call site is not blocked while waiting for the called code to finish, and instead the calling thread is notified when the reply arrives.

Chapter 4

Andriod Studio

4.1 Android StudiO:

It is the official[7] integrated development environment (IDE) for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development.[8] It is available for download on Windows, macOS and Linux based operating systems.[9] It is a replacement for the Eclipse Android Development Tools (E-ADT) as the primary IDE for native Android application development.

Android Studio was announced on May 16, 2013, at the Google I/O conference. It was in early access preview stage starting from version 0.1 in May 2013, then entered beta stage starting from version 0.8 which was released in June 2014.[10] The first stable build was released in December 2014, starting from version 1.0.[11] At the end of 2015, Google dropped support for Eclipse ADT, making Android Studio the only officially supported IDE for Android development.[12]

On May 7, 2019, Kotlin replaced Java as Google's preferred language for Android app development.[13] Java is still supported, as is C++.[14]

4.2 Features

The following features are provided in the current stable version:[15][16]

- Gradle-based build support
- Android-specific refactoring and quick fixes
- Lint tools to catch performance, usability, version compatibility and other problems
- ProGuard integration and app-signing capabilities
- Template-based wizards to create common Android designs and components
- A rich layout editor that allows users to drag-and-drop UI components, option to preview layouts on multiple screen configurations[17]
- Support for building Android Wear apps

- Built-in support for Google Cloud Platform, enabling integration with Firebase Cloud Messaging (Earlier 'Google Cloud Messaging') and Google App Engine[18]
- Android Virtual Device (Emulator) to run and debug apps in the Android studio.

Android Studio supports all the same programming languages of IntelliJ (and CLion) e.g. Java, C++, and more with extensions, such as Go;[19] and Android Studio 3.0 or later supports Kotlin[20] and "all Java 7 language features and a subset of Java 8 language features that vary by platform version."[21] External projects backport some Java 9 features.[22] While IntelliJ states that Android Studio supports all released Java versions, and Java 12, it's not clear to what level Android Studio supports Java versions up to Java 12 (the documentation mentions partial Java 8 support). At least some new language features up to Java 12 are usable in Android.[23]

Once an app has been compiled with Android Studio, it can be published on the Google Play Store. The application has to be in line with the Google Play Store developer content policy.

Version history

Table no 2. of Android Studio's major releases:[24]

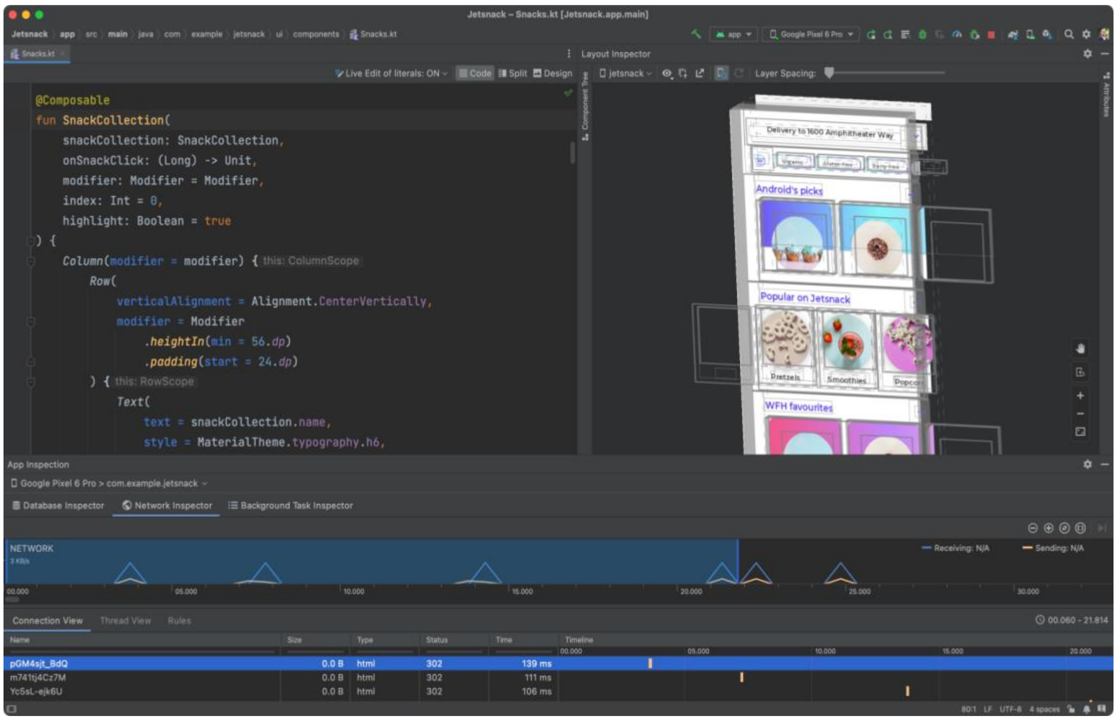
Version	**Release date**
1.0	December 2014
1.1	February 2015
1.2	April 2015
1.3	July 2015
1.4	September 2015
1.5	November 2015
2.0	April 2016
2.1	April 2016
2.2	September 2016
2.3	March 2017
3.0	October 2017

Version	Release date
3.1	March 2018
3.2	September 2018
3.3	January 2019
3.4	April 2019[25]
3.5	August 2019
3.6	February 2020
4.0	May 2020
4.1	Oct 2020[26]
4.2	May 2021[27]
Arctic Fox (2020.3.1)	July 2021[28]
Bumblebee (2021.1.1)	January 2022[29]
Chipmunk (2021.2.1)	May 2022[30]
Dolphin (2021.3.1)	September 2022[31]
Electric Eel (2022.1.1)	January 2023[32]
Flamingo (2022.2.1)	April 2023[33]
Giraffe (2022.3.1)	TBD
Hedgehog (2023.1.1)	TBD

features includes requirements for IDE + Android SDK + Android Emulator.[35]

- Windows: x86_64 CPU architecture; 2nd generation Intel Core or newer, or AMD CPU with support for a Windows Hypervisor;
- macOS: ARM-based chips, or 2nd generation Intel Core or newer with support for Hypervisor.Framework;
- Linux: x86_64 CPU architecture; 2nd generation Intel Core or newer, or AMD processor with support for AMD Virtualization (AMD-V) and SSSE3;
- Windows: CPU with UG (unrestricted guest) support;
- Intel Hardware Accelerated Execution Manager (**HAXM**) 6.2.1 or later (HAXM 7.2.0 or later recommended).

- The use of hardware acceleration has additional requirements on Windows and Linux:
- Intel processor on Windows or Linux: Intel processor with support for Intel VT-x, Intel EM64T (Intel 64), and Execute Disable (XD) Bit functionality;
- AMD processor on Linux: AMD processor with support for AMD Virtualization (AMD-V) and Supplemental Streaming SIMD Extensions 3 (SSSE3);
- AMD processor on Windows: Android Studio 3.2 or higher and Windows 10 April 2018 release or higher for Windows Hypervisor Platform (WHPX) functionality.



```
@Composable
fun SnackCollection(
    snackCollection: SnackCollection,
    onSnackClick: (Long) -> Unit,
    modifier: Modifier = Modifier,
    index: Int = 0,
    highlight: Boolean = true
) {
    Column(modifier = modifier) {
        Row(
            verticalAlignment = Alignment.CenterVertically,
            modifier = Modifier
                .heightIn(min = 56.dp)
                .padding(start = 24.dp)
        ) {
            Text(
                text = snackCollection.name,
                style = MaterialTheme.typography.h6,
```

Name	Size	Type	Status	Time
pGM4sjt_BdQ	0.0 B	html	302	139 ms
m741tj4Cz7M	0.0 B	html	302	111 ms
Yc5sL-ejk6U	0.0 B	html	302	106 ms

Fig 4.Andriod Studio Image

4.3 XML :

It stands for Extensible Markup Language. XML is a markup language much like HTML used to describe data. It is derived from Standard Generalized Markup Language(SGML). Basically, the XML tags are not predefined in XML. We need to implement and define the tags in XML. XML tags define the data and used to store and organize data. It's easily scalable and simple to develop. In Android, the XML is used to implement UI-related data, and it's a lightweight markup language that

]

doesn't make layout heavy. XML only contains tags, while implementing they need to be just invoked.

Fig 5.XML image

The simple syntax of XML is

<tag_name>Hello World!</tag_name>

So in this article, there is a deep discussion on how to learn and understand what XML is for Android Development.

Basics of User Interface(UI)

Basically in Android XML is used to implement the UI-related data. So understanding the core part of the UI interface with respect to XML is important. The User Interface for an Android App is built as the hierarchy of main **layouts, widgets**. The layouts are **ViewGroup** objects or containers that control how the child view should be positioned on the screen. **Widgets** here are view objects, such as Buttons and text boxes.

Output UI:

In the above example of XML file, There are 2 view groups one is LinearLayout and another is RelativeLayout and the TextView1 and TextView2 is child widgets under the ViewGroup1 that is LinearLayout. EditText1, EditText2, and Button are the child widgets under ViewGroup2 that is RelativeLayout. ViewGroup2(RelativeLayout) is nested under the ViewGroup1 which produces the following hierarchy.

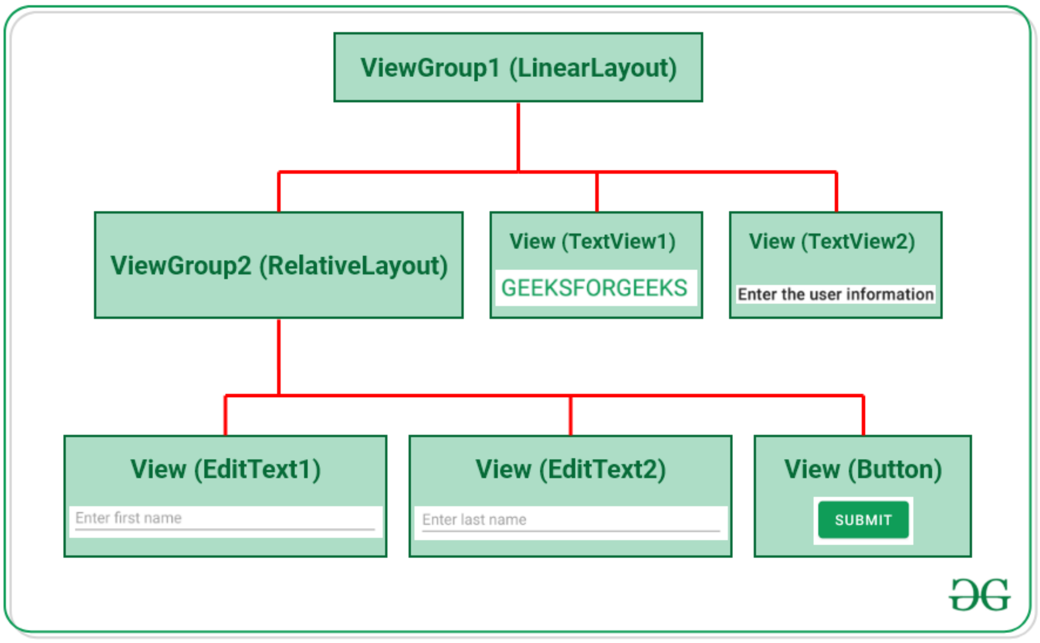


Fig 6 .ViewGroup

From the hierarchy, it's been observed that every widget like EdtText, TextView, or Button is one of the View. These Views are contained inside the ViewGroup, like RelativeLayout, LinearLayout, FrameLayout, etc.

Different Types of XML Files Used in Android Studio

Different XML files serve different purposes in Android Studio. The list of various XML files in Android Studio with their purposes is discussed below.

4.3.1. Layout XML files in android

The Layout XML files are responsible for the actual User Interface of the application. It holds all the widgets or views like Buttons, TextViews, EditTexts, etc. which are defined under the ViewGroups. **The Location of the layout files in Android is:**

app -> src -> main -> res -> layout

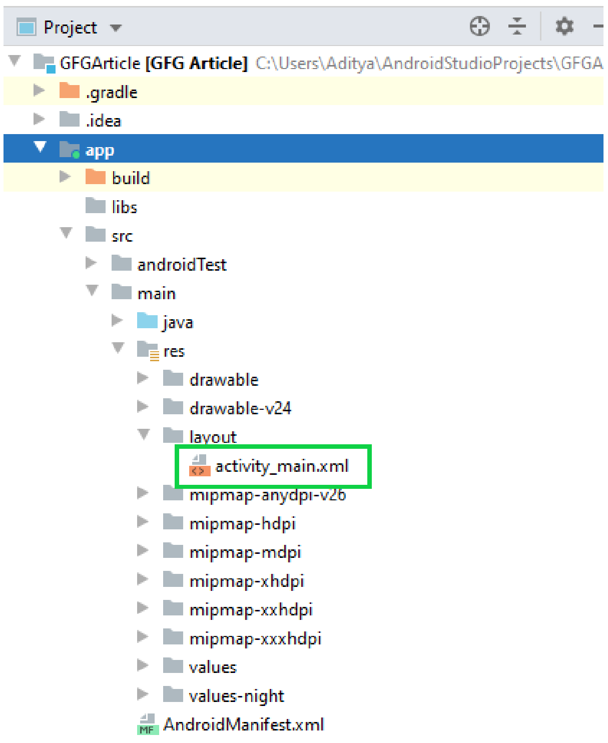
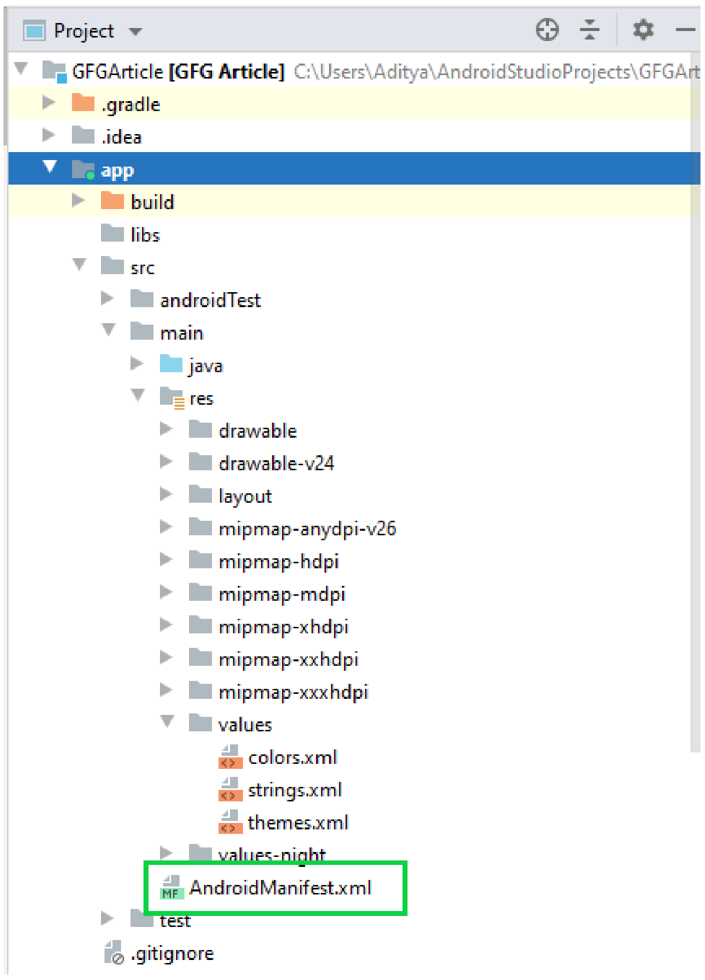


Fig 7 . XML layout

The folder contains the layout files for respective activity, fragments. The basic layout of the is as follows for activity_main.xml:

4.3.2. AndroidManifest.xml file

This file describes the essential information about the application's, like the application's package names which matches code's namespaces, a component of the application like activities, services, broadcast receivers, and content providers. Permission required by the user for the application features also mentioned in this XML file. **Location of the Android Manifest.xml file:** ***app -> src -> main***



```
Project
GFGArticle [GFG Article] C:\Users\Aditya\AndroidStudioProjects\GFGArt
  .gradle
  .idea
  app
    build
    libs
    src
      androidTest
      main
        java
        res
          drawable
          drawable-v24
          layout
          mipmap-anydpi-v26
          mipmap-hdpi
          mipmap-mdpi
          mipmap-xhdpi
          mipmap-xxhdpi
          mipmap-xxxhdpi
          values
            colors.xml
            strings.xml
            themes.xml
          values-night
        AndroidManifest.xml
      test
    .gitignore
```

Fig 9.Andriod MANIFEST File

4.3.3. Strings.xml file

This file contains texts for all the TextViews widgets. This enables reusability of code, and also helps in the localization of the application with different languages. The strings defined in these files can be used to replace all hardcoded text in the entire application. **Location of the strings.xml file**

app -> src -> main -> res -> values

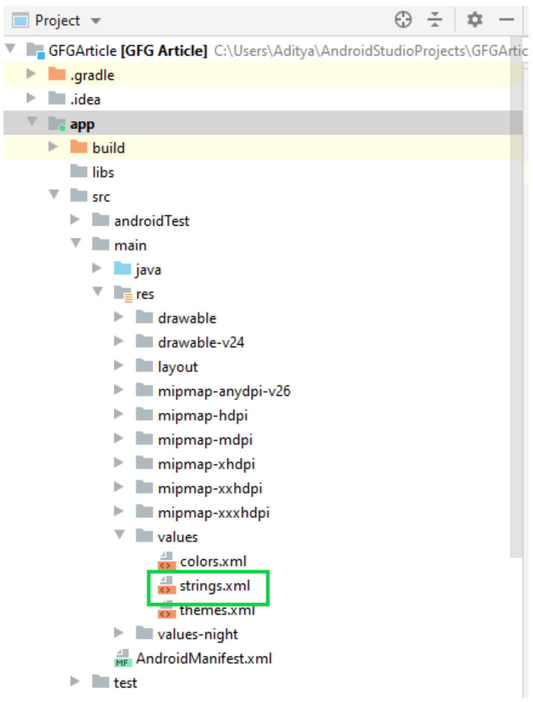


Fig 9. String XML

A typical code inside strings.xml looks like this:

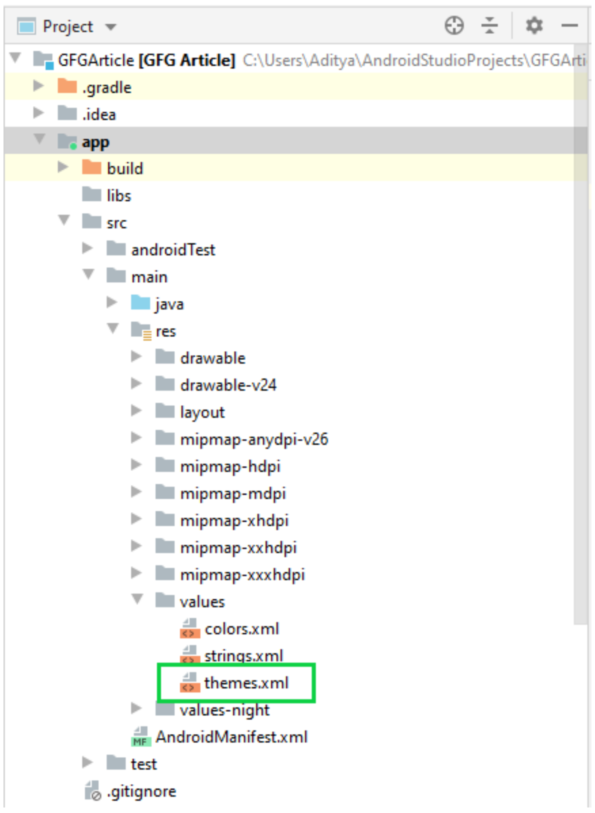
- XML

```
<resources>
    <string name="app_name">GFG Article</string>
    <string name="gfg_heading">GEEKSFORGEEKS</string>
    <string name="settings_label">settings</string>
</resources>
```

4.3.4. Themes.xml file

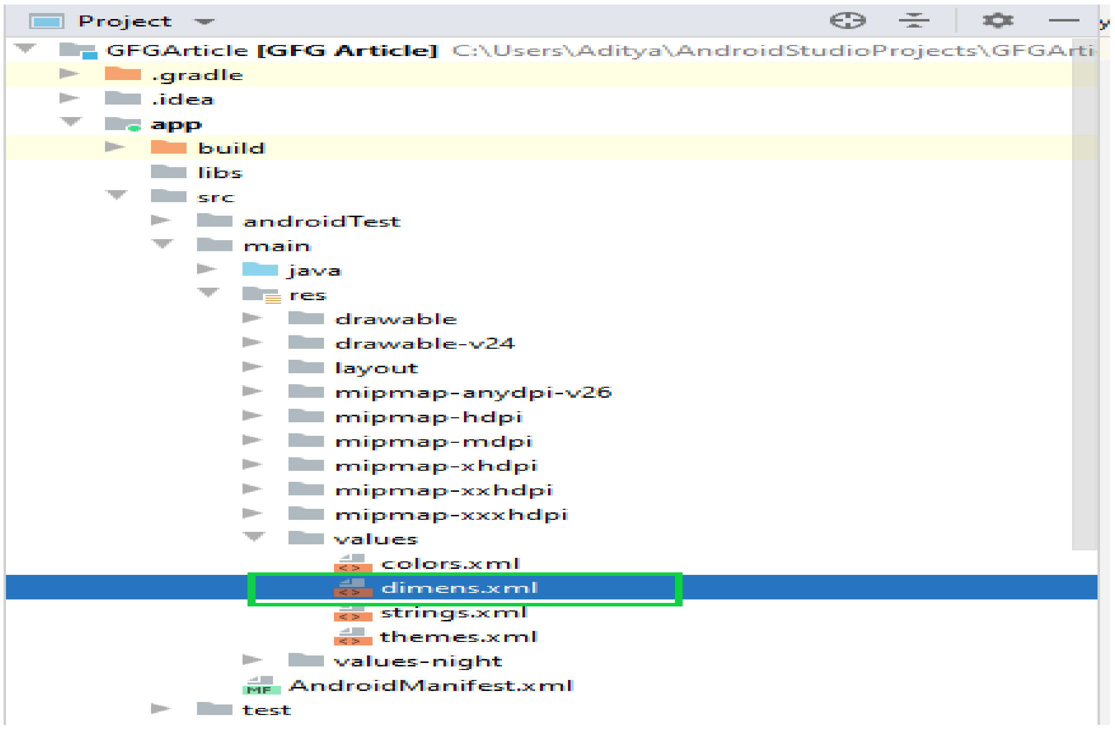
This file defines the base theme and customized themes of the application. It also used to define styles and looks for the UI(User Interface) of the application. By defining styles we can customize how the views or widgets look on the User Interface. **Location of styles.xml file**

app -> src -> main -> res -> values



```
Project
GFGArticle [GFG Article] C:\Users\Aditya\AndroidStudioProjects\GFGArti
  .gradle
  .idea
  app
    build
    libs
    src
      androidTest
      main
        java
        res
          drawable
          drawable-v24
          layout
          mipmap-anydpi-v26
          mipmap-hdpi
          mipmap-mdpi
          mipmap-xhdpi
          mipmap-xxhdpi
          mipmap-xxxhdpi
          values
            colors.xml
            strings.xml
            themes.xml
          values-night
        AndroidManifest.xml
      test
    .gitignore
```

Fig.10 Themes XML



```
Project
GFGArticle [GFG Article] C:\Users\Aditya\AndroidStudioProjects\GFGArti
  .gradle
  .idea
  app
    build
    libs
    src
      androidTest
      main
        java
        res
          drawable
          drawable-v24
          layout
          mipmap-anydpi-v26
          mipmap-hdpi
          mipmap-mdpi
          mipmap-xhdpi
          mipmap-xxhdpi
          mipmap-xxxhdpi
          values
            colors.xml
            dimens.xml
            strings.xml
            themes.xml
          values-night
        AndroidManifest.xml
      test
```

Fig 11 Dimens.Xml File.

Chapter 5

Java Language

Java is one of the powerful general-purpose programming languages, created in 1995 by Sun Microsystems (now owned by Oracle). Java is Object-Oriented. However, it is not considered as pure object-oriented as it provides support for primitive data types (like int, char, etc). Java syntax is similar to C/C++. But Java does not provide low-level programming functionalities like pointers. Also, Java code is always written in the form of classes and objects. Android heavily relies on the Java programming language all the SDKs required to build for android applications use the standard libraries of Java. If one is coming from a traditional programming background like C, C++, Java is easy to learn. So in this discussion, there is a complete guide to learn Java specifically considering Android App Development.

Fig . 12 Java Image Logo

So in this article, we have covered the following things:

4. **Basics of Java**
5. **Decision Making Statements in Java**
6. **Type Conversion in Java**
7. **Comments in Java**
8. **Operators in Java**
9. **Strings in Java**
10. **Object-Oriented Programming Concepts in Java**
11. **Exception Handling in Java**

12. **Interfaces and Abstract Classes**
13. **Essential collections in Java required for Android Development**
14. **Miscellaneous**
15. **Complete Java Tutorial**

5.1 Step-by-Step Guide to Learn Java for Android App Development

Basics of Java

- How to start learning Java – understand the core introduction of the Java programming language.
- Setting up the environment – Setup IDE for writing programs in Java.
- The Hello World Example – The first Hello World program in Java.
- Java Class File – Basic entry point of Java programming, which is writing the main class.
- Java Identifiers – In Java, an identifier can be a class name, method name, variable name, or label.
- Data types in Java – Get to know what types of data types are supported by the Java programming language.
- Variables in Java – A variable is a name given to a memory location. It is the basic unit of storage in a program.
- Scope of Variables – The scope of a variable is the part of the program where the variable is accessible.
- Blank Final in Java – A final variable in Java can be assigned a value only once. We can assign a value either in the declaration or later.

Decision Making Statements in Java

- Decision Making in Java (if, if-else, switch, break, continue, jump) – A programming language uses control statements to control the flow of execution of a program based on certain conditions.
- Switch Statement in Java – The switch statement is a multi-way branch statement. It provides an easy way to dispatch execution to different parts of code based on the value of the expression.

- Loops in Java – Looping in programming languages is a feature that facilitates the execution of a set of instructions/functions repeatedly while some conditions are evaluated to be true.
- For-each loop in Java – For-each is another array traversing technique like for loop, while loop, do-while loop is introduced in Java5.

Type Conversion in Java

- Type conversion in Java with Examples – If the data types are compatible, then Java will perform the conversion automatically known as Automatic Type Conversion, and if not, then they need to be cast or converted explicitly.

Comments in Java

- Comments in Java – Comments take part in making the program become more human-readable by placing the details of code involved and proper use of comments makes maintenance easier and finding bugs easier.

Operators in Java

- Operators in Java – Java provides many types of operators which can be used according to the need. They are classified based on the functionality they provide.

Strings in Java

- String class in Java | Set 1 – String is a sequence of characters. In Java, objects of strings are immutable, which means constant and cannot be changed once created.
- StringBuffer class in Java – StringBuffer is a peer class of String that provides much of the functionality of strings.
- StringBuilder Class in Java with Examples – The StringBuilder in Java represents a mutable sequence of characters.

Object-Oriented Programming Concepts in Java

- Classes and Objects in Java – The basic OOPs components Class and Object in the java programming language.
- Different ways to create objects in Java – Get to know the various ways of creating objects in Java.
- Inheritance in Java – It is the mechanism in Java by which one class is allowed to inherit the features(fields and methods) of another class.

- Encapsulation in Java – Encapsulation is defined as the wrapping up of data under a single unit.
- Abstraction in Java – Data Abstraction is a property by virtue of which only the essential details are displayed to the user.
- Access Modifiers in Java – As the name suggests, access modifiers in Java help to restrict the scope of a class, constructor, variable, method, or data member.
- 'this' reference in Java – 'this' is a reference variable that refers to the current object.
- Overloading in Java – Overloading allows different methods to have the same name, but different signatures of methods.
- Overriding in Java – Overriding is a feature that allows a subclass or child class to provide a specific implementation of a method that is already provided by one of its super-classes or parent classes.
- Object class in Java – Object class is present in the java.lang package. Every class in Java is directly or indirectly derived from the Object class.
- Static class in Java – Some classes can be made static in Java. Java supports Static Instance Variables, Static Methods, Static Block, and Static Classes.

Exception Handling in Java

- Exceptions in Java – An exception is an unwanted or unexpected event that occurs during the execution of a program i.e at run time.
- Types of Exception in Java with Examples – Java also allows users to define their own exceptions.

Interfaces and Abstract Classes

- Interfaces in Java – Like a class, an interface can have methods and variables, but the methods declared in an interface are by default abstract.
- Access specifier of methods in interfaces – All methods in an interface are public, even if we do not specify public with method names. Also, data fields are public static final even if we do not mention them in field names.
- Access specifiers for classes or interfaces in Java – Methods and data members of a class/interface can have one of the following four access specifiers.
- Abstract Classes in Java – Java, a separate keyword abstract is used to make a class abstract.

- Difference between Abstract Class and Interface in Java – Get to know the differences between the interfaces and abstract classes.
- Anonymous Inner Class in Java – It is an inner class without a name and for which only a single object is created.

Essential collections in Java required for Android Development

- ArrayList in Java – ArrayList is a part of the collection framework and is present in the java.util package. It provides us with dynamic arrays in Java.
- HashMap in Java with Examples – It stores the data in (Key, Value) pairs, and you can access it via an index of another type.

Miscellaneous

- Java Naming Conventions – Naming conventions must be followed while developing software in Java for good maintenance and readability of code.
- Generics in Java – Generics mean parameterized types. The idea is to allow types (Ingers, strings, … etc, and user-defined types) to be a parameter for methods, classes, and interfaces.
- Annotations in Java – Annotations are used to provide supplemental information about a program.
- Lambda Expressions in Java 8 – Lambda expressions basically express instances of functional interfaces (An interface with a single abstract method is called a functional interface.

.

.Chapter 6

Java Development Kit

The Java Development Kit (**JDK**) is a distribution of Java Technology by Oracle Corporation. It implements the Java Language Specification (**JLS**) and the Java Virtual Machine Specification (**JVMS**) and provides the Standard Edition (**SE**) of the Java Application Programming Interface (**API**). It is derivative of the community driven OpenJDK which Oracle stewards.[5] It provides software for working with Java applications. Examples of included software are the virtual machine, a compiler, performance monitoring tools, a debugger, and other utilities that Oracle considers useful for a Java programmer.

Oracle have released the current version of the software under the Oracle No-Fee Terms and Conditions (**NFTC**) license. Oracle release binaries for the x86-64 architecture for Windows, macOS, and Linux based operating systems, and for the aarch64 architecture for macOS and Linux. Previous versions have supported the Oracle Solaris operating system and SPARC architecture.
Oracle's primary implementation of the JVMS is known as the HotSpot (virtual machine).

6.1 JDK contents

The JDK has as its primary components a collection of programming tools, including:

- appletviewer – this tool can be used to run and debug Java applets without a web browser
- apt – the annotation-processing tool[6]
- extcheck – a utility that detects JAR file conflicts
- idlj – the IDL-to-Java compiler. This utility generates Java bindings from a given Java IDL file.
- jabswitch – the Java Access Bridge. Exposes assistive technologies on Microsoft Windows systems.
- java – the loader for Java applications. This tool is an interpreter and can interpret the class files generated by the javac compiler. Now a single launcher is used for both development and deployment. The old deployment launcher,

jre, no longer comes with Sun JDK, and instead it has been replaced by this new java loader.

- javac – the Java compiler, which converts source code into Java bytecode
- javadoc – the documentation generator, which automatically generates documentation from source code comments
- jar – the archiver, which packages related class libraries into a single JAR file. This tool also helps manage JAR files.
- javafxpackager – tool to package and sign JavaFX applications
- jarsigner – the jar signing and verification tool
- javah – the C header and stub generator, used to write native methods
- javap – the class file disassembler
- javaws – the Java Web Start launcher for JNLP applications
- JConsole – Java Monitoring and Management Console
- jdb – the debugger
- jhat – Java Heap Analysis Tool (experimental)
- jinfo – This utility gets configuration information from a running Java process or crash dump. (experimental)
- jmap Oracle jmap - Memory Map– This utility outputs the memory map for Java and can print shared object memory maps or heap memory details of a given process or core dump. (experimental)
- jmc – Java Mission Control
- jpackage – a tool for generating self-contained application bundles. (experimental)
- jps – Java Virtual Machine Process Status Tool lists the instrumented HotSpot Java Virtual Machines (JVMs) on the target system. (experimental)
- jrunscript – Java command-line script shell.
- jshell - a read–eval–print loop, introduced in Java 9.
- jstack – utility that prints Java stack traces of Java threads (experimental)
- jstat – Java Virtual Machine statistics monitoring tool (experimental)
- jstatd – jstat daemon (experimental)
- keytool – tool for manipulating the keystore
- pack200 – JAR compression tool

- policytool – the policy creation and management tool, which can determine policy for a Java runtime, specifying which permissions are available for code from various sources.
- VisualVM – visual tool integrating several command-line JDK tools and lightweight[*clarification needed*] performance and memory profiling capabilities (no longer included in JDK 9+)
- wsimport – generates portable JAX-WS artifacts for invoking a web service.
- xjc – Part of the Java API for XML Binding (JAXB) API. It accepts an XML schema and generates Java classes.

Experimental tools may not be available in future versions of the JDK.

The JDK also comes with a complete Java Runtime Environment (JRE), usually called a *private* runtime, due to the fact that it is separated from the "regular" JRE and has extra contents. It consists of a Java virtual machine and all of the class libraries present in the production environment, as well as additional libraries only useful to developers, such as the internationalization libraries and the IDL libraries.

Fig 13. JDK LOGO

Chapter 7

Andriod APP

Android is an **operating system** that is built basically for Mobile phones. It is based on the Linux Kernel and other open-source software and is developed by **Google**. It is used for touchscreen mobile devices such as smartphones and tablets. But nowadays these are used in Android Auto cars, TV, watches, camera, etc. It has been one of the best-selling OS for smartphones. Android OS was developed by **Android Inc**. which Google bought in 2005. Various applications (apps) like games, music player, camera, etc. are built for these smartphones for running on Android. **Google Play store** features more than 3.3 million apps. The app is developed on an application known as **Android Studio.** These executable apps are installed through a bundle or package called **APK(Android Package Kit)**.

7.1 Android Fundamentals

1. Android Programming Languages

In Android, basically, programming is done in two languages **JAVA or C++** and **XML(Extension Markup Language)**. Nowadays KOTLIN is also preferred. The XML file deals with the design, presentation, layouts, blueprint, etc (as a front-end) while the JAVA or KOTLIN deals with the working of buttons, variables, storing, etc (as a back-end).

2. Android Components

The App components are the building blocks of Android. Each component has its own role and life cycles i.e from launching of an app till the end. Some of these components depend upon others also. Each component has a definite purpose. The four major app components are:

- Activities
- Services
- Broadcast Receivers:
- Content Provider:

Activities: It deals with the UI and the user interactions to the screen. In other words, it is a User Interface that contains activities. These can be one or more depending upon the App. It starts when the application is launched. At least one activity is always present which is known as MainActivity. The activity is implemented through the following.

Syntax:

```
public class MainActivity extends Activity{
  // processes
}
```

To know more Activities please refer to this article: Introduction to Activities in Android

Services: Services are the background actions performed by the app, these might be long-running operations like a user playing music while surfing the Internet. A service might need other sub-services so as to perform specific tasks. The main purpose of the Services is to provide non-stop working of the app without breaking any interaction with the user.

Syntax:

```
public class MyServices extends Services{
  // code for the services
}
```

To know more Services please refer to this article: Services in Android with Example

Broadcast Receivers: A Broadcast is used to respond to messages from other applications or from the System. For example, when the battery of the phone is low, then the Android OS fires a Broadcasting message to launch the Battery Saver function or app, after receiving the message the appropriate action is taken by the app. Broadcast Receiver is the subclass of BroadcastReceiver class and each object is represented by Intent objects.

Syntax:

```
public class MyReceiver extends BroadcastReceiver{
   public void onReceive(context,intent){
 }
```

To know more Broadcast Receivers please refer to this article: Broadcast Receiver in Android With Example

Content Provider: Content Provider is used to transferring the data from one application to the others at the request of the other application. These are handled by the class ContentResolver class. This class implements a set of APIs(Application Programming Interface) that enables the other applications to perform the transactions. Any Content Provider must implement the Parent Class of ContentProvider class.

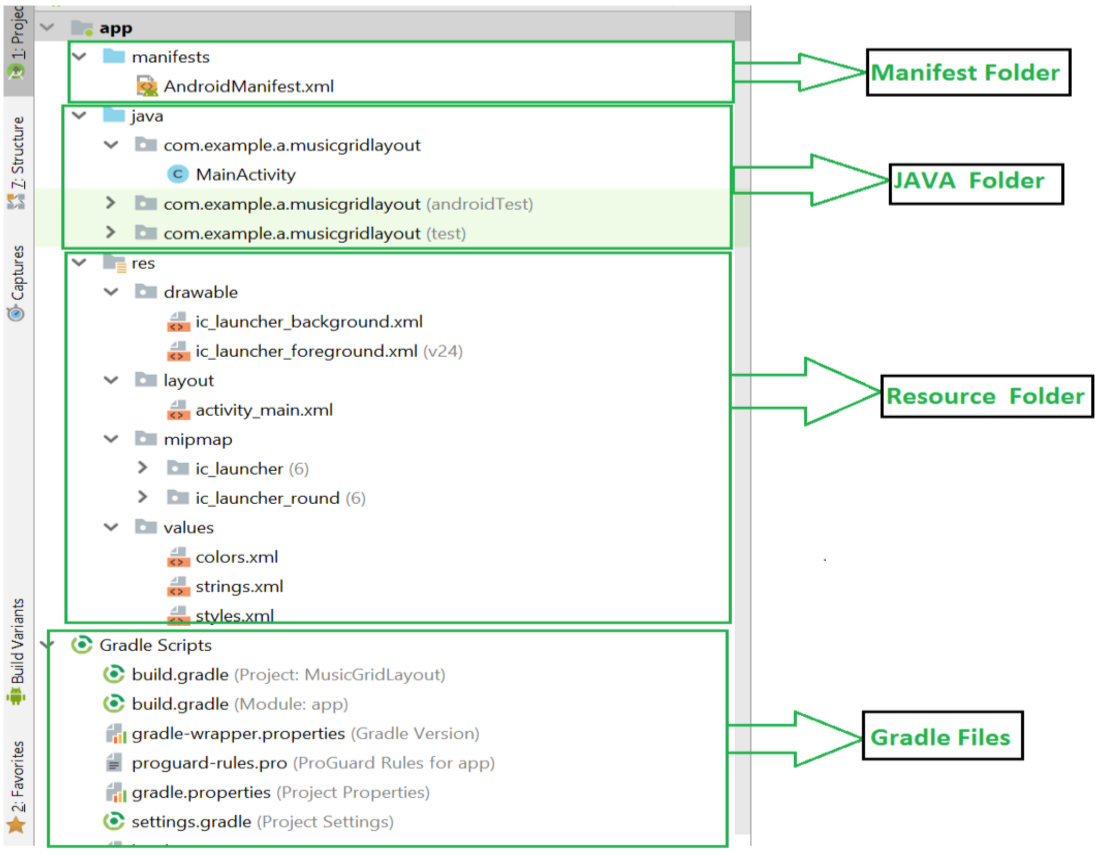
Syntax:

```
public class MyContentProvider extends ContentProvider{
  public void onCreate()
  {}
}
```

To know more Content Provider please refer to this article: Content Providers in Android with Example

3. Structural Layout Of Android Studio

The basic structural layout of Android Studio is given below:



```
app
  manifests
    AndroidManifest.xml
  java
    com.example.a.musicgridlayout
      MainActivity
    com.example.a.musicgridlayout (androidTest)
    com.example.a.musicgridlayout (test)
  res
    drawable
      ic_launcher_background.xml
      ic_launcher_foreground.xml (v24)
    layout
      activity_main.xml
    mipmap
      ic_launcher (6)
      ic_launcher_round (6)
    values
      colors.xml
      strings.xml
      styles.xml
Gradle Scripts
  build.gradle (Project: MusicGridLayout)
  build.gradle (Module: app)
  gradle-wrapper.properties (Gradle Version)
  proguard-rules.pro (ProGuard Rules for app)
  gradle.properties (Project Properties)
  settings.gradle (Project Settings)
```

Figure 14 .represents the various structure of an app.

Manifest Folder: Android Manifest is an XML file that is the root of the project source set. It describes the essential information about the app and the Android build tools, the Android Operating System, and Google Play. It contains the permission that an app might need in order to perform a specific task. It also contains the Hardware and the Software features of the app, which determines the compatibility of an app on the Play Store. It also includes special activities like services, broadcast receiver, content providers, package name, etc.

Java Folder: The **JAVA folder** consists of the java files that are required to perform the background task of the app. It consists of the functionality of the buttons, calculation, storing, variables, toast(small popup message), programming function, etc. The number of these files depends upon the type of activities created.

Resource Folder: The res or Resource folder consists of the various resources that are used in the app. This consists of sub-folders like drawable, layout, mipmap, raw, and values. The drawable consists of the images. The layout consists of the XML files that define the user interface layout. These are stored in res.layout and are accessed as R.layout class. The raw consists of the Resources files like audio files or music files, etc. These are accessed through R.raw.filename. values are used to store the hardcoded strings(considered safe to store string values) values, integers, and colors. It consists of various other directories like:

- **R.array :arrays.xml for resource arrays**
- **R.integer : integers.xml for resource integers**
- **R.bool : bools.xml for resource boolean**
- **R.color :colors.xml for color values**
- **R.string : strings.xml for string values**
- **R.dimen : dimens.xml for dimension values**
- **R.style : styles.xml for styles**

Gradle Files: Gradle is an advanced toolkit, which is used to manage the build process, that allows defining the flexible custom build configurations. Each build configuration can define its own set of code and resources while reusing the parts common to all versions of your app. The Android plugin for Gradle works with the build toolkit to provide processes and configurable settings that are specific to building and testing Android applications. Gradle and the Android plugin run independently of Android Studio. This means that you can build your Android apps from within Android Studio. The flexibility of the Android build system enables you to perform custom build configurations without modifying your app’s core source files.

Basic Layout Can be defined in a tree structure as:

```
Project/
  app/
    manifest/
      AndroidManifest.xml
  java/
    MyActivity.java
    res/
      drawable/
        icon.png
        background.png
      drawable-hdpi/
        icon.png
        background.png
      layout/
        activity_main.xml
        info.xml
      values/
        strings.xml
```

4. Lifecycle of Activity in Android App

The **Lifecycle** of Activity in Android App can be shown through this diagram:

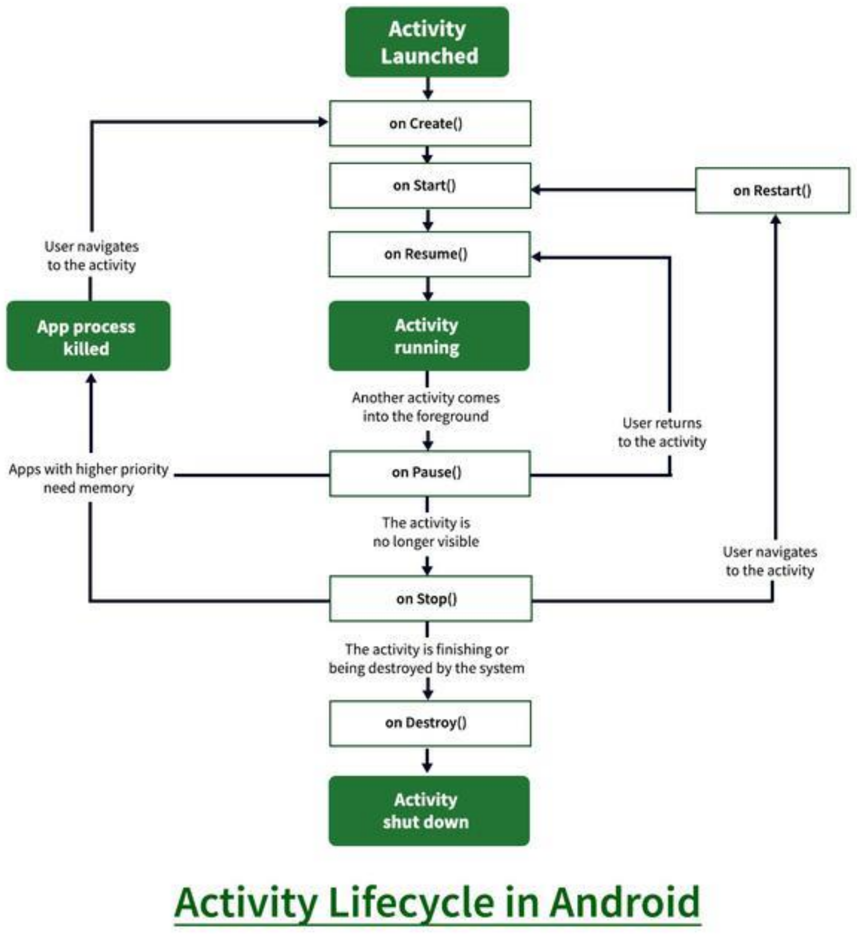

Activity Launched
on Create()
on Start()
on Restart()
User navigates to the activity
on Resume()
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
on Pause()
The activity is no longer visible
User navigates to the activity
on Stop()
The activity is finishing or being destroyed by the system
on Destroy()
Activity shut down
Activity Lifecycle in Android

Fig 15.Andriod Lifecycle

States of Android Lifecycle:

1. **OnCreate:** This is called when activity is first created.
2. **OnStart:** This is called when the activity becomes visible to the user.
3. **OnResume:** This is called when the activity starts to interact with the user.
4. **OnPause:** This is called when activity is not visible to the user.
5. **OnStop:** This is called when activity is no longer visible.
6. **OnRestart:** This is called when activity is stopped, and restarted again.
7. **OnDestroy:** This is called when activity is to be closed or destroyed.

Fig . Andriod logo.

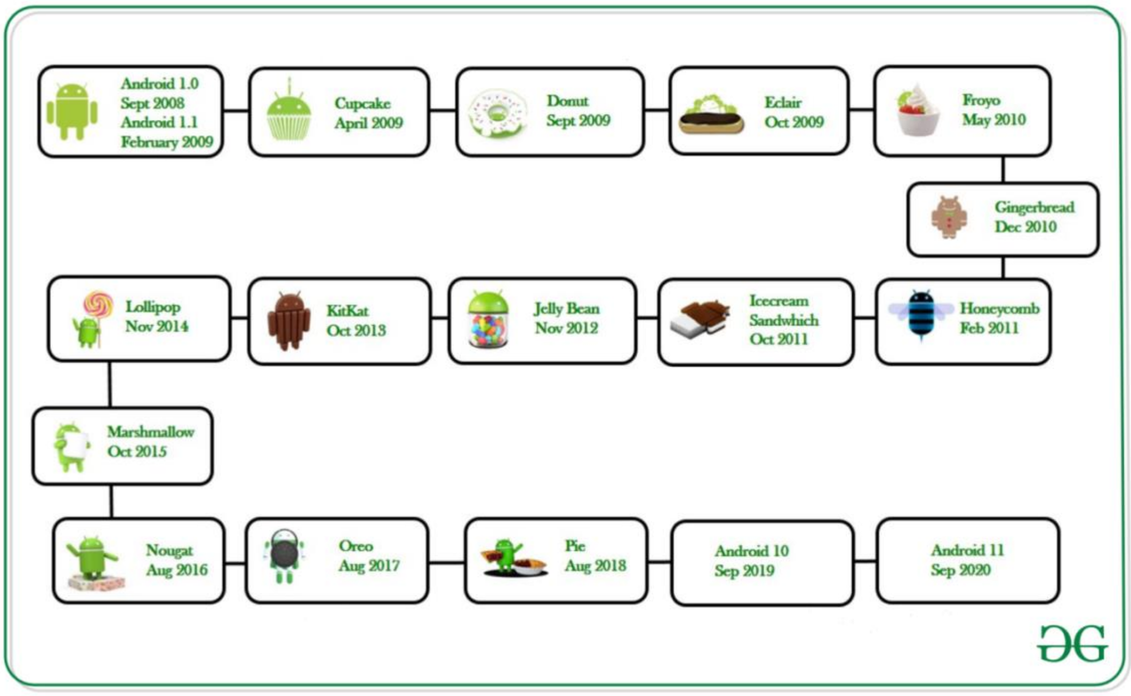


Fig 17 .Vesrion

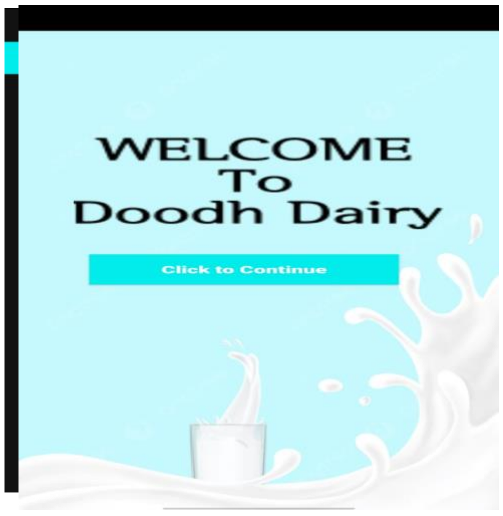
Chapter 8

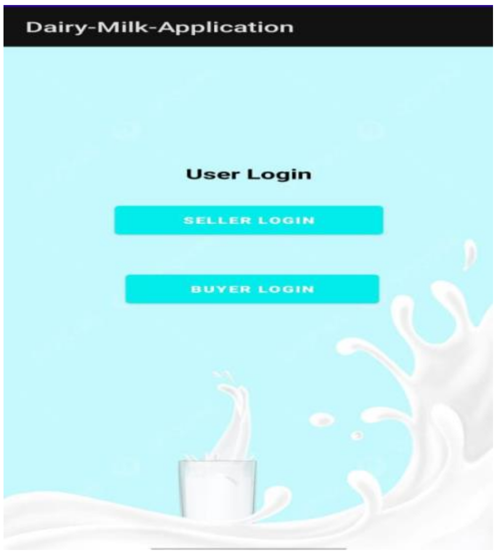
Result

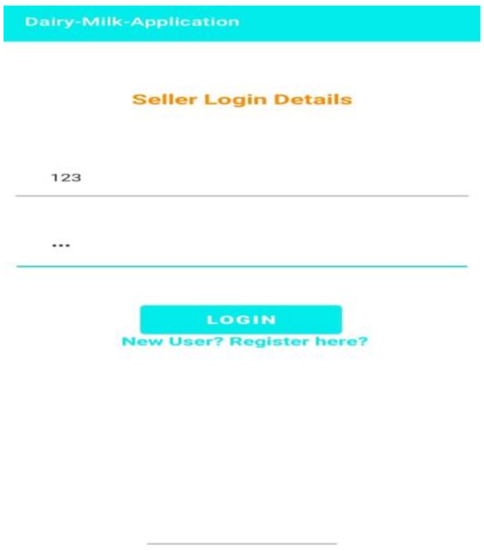
Result:

The result of the project is fully functional Android application that allows Dairy owners and customers to maintain their daily dairy data. However, this project provides a strong foundation for future improvements and enhancements. Overall this project serves as a useful tool for milk sellers and customers to store their data in the on-line mode without the use of hardcopies.

Result Image

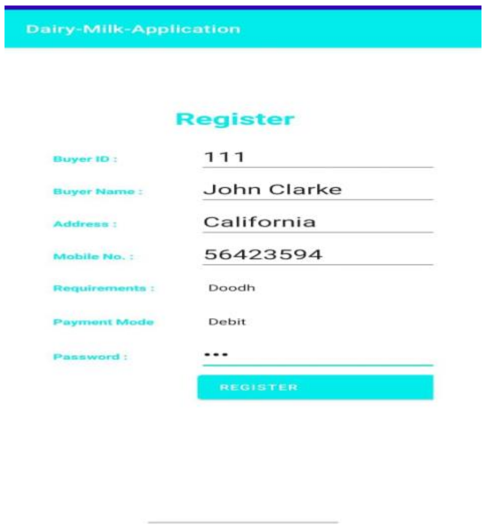




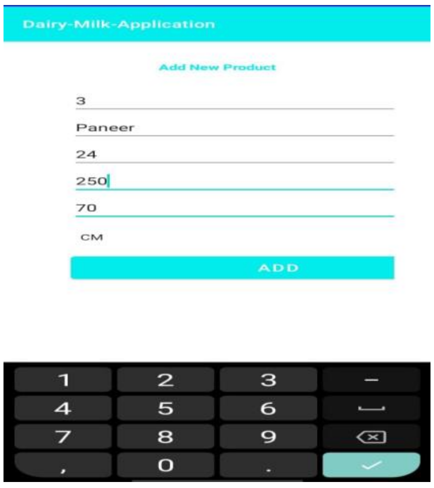


123

...



Buyer ID :	111
Buyer Name :	John Clarke
Address :	California
Mobile No. :	56423594
Requirements :	Doodh
Payment Mode	Debit
Password :	•••


Dairy-Milk-Application

Add New Product

3

Paneer

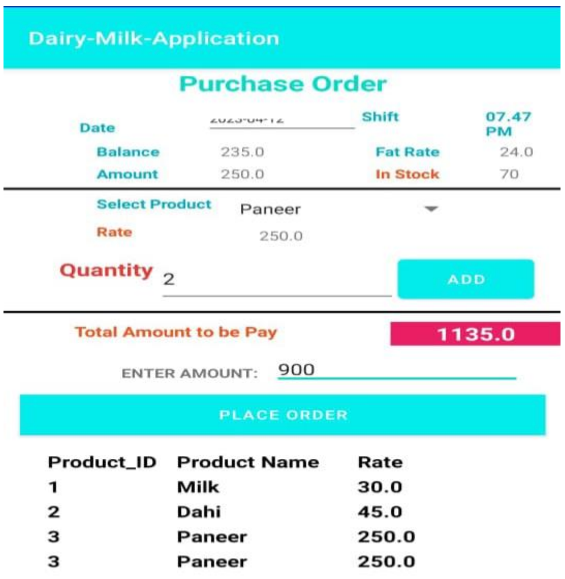
24

250

70

CM

ADD


Dairy-Milk-Application

Purchase Order

Date 2023-04-12 Shift 07.47 PM

Balance Amount 235.0 250.0

Fat Rate 24.0

In Stock 70

Select Product Paneer

Rate 250.0

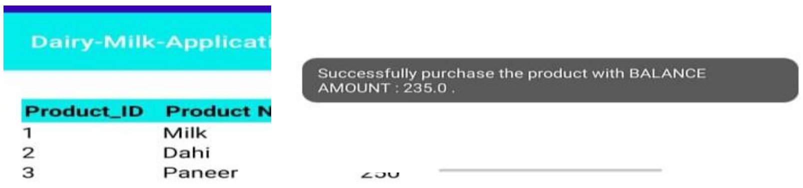
Quantity 2 ADD

Total Amount to be Pay 1135.0

ENTER AMOUNT: 900

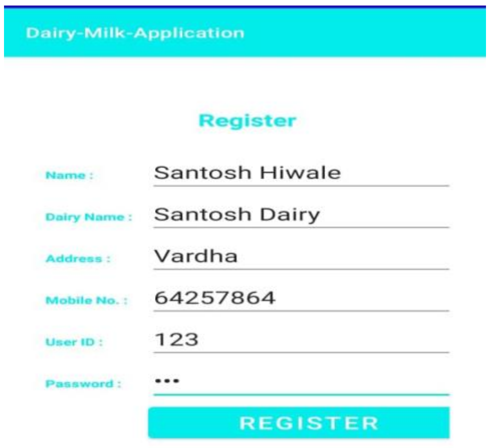
PLACE ORDER

Product_ID	Product Name	Rate
1	Milk	30.0
2	Dahi	45.0
3	Paneer	250.0
3	Paneer	250.0


Dairy-Milk-Applicat

Product_ID	Product N
1	Milk
2	Dahi
3	Paneer

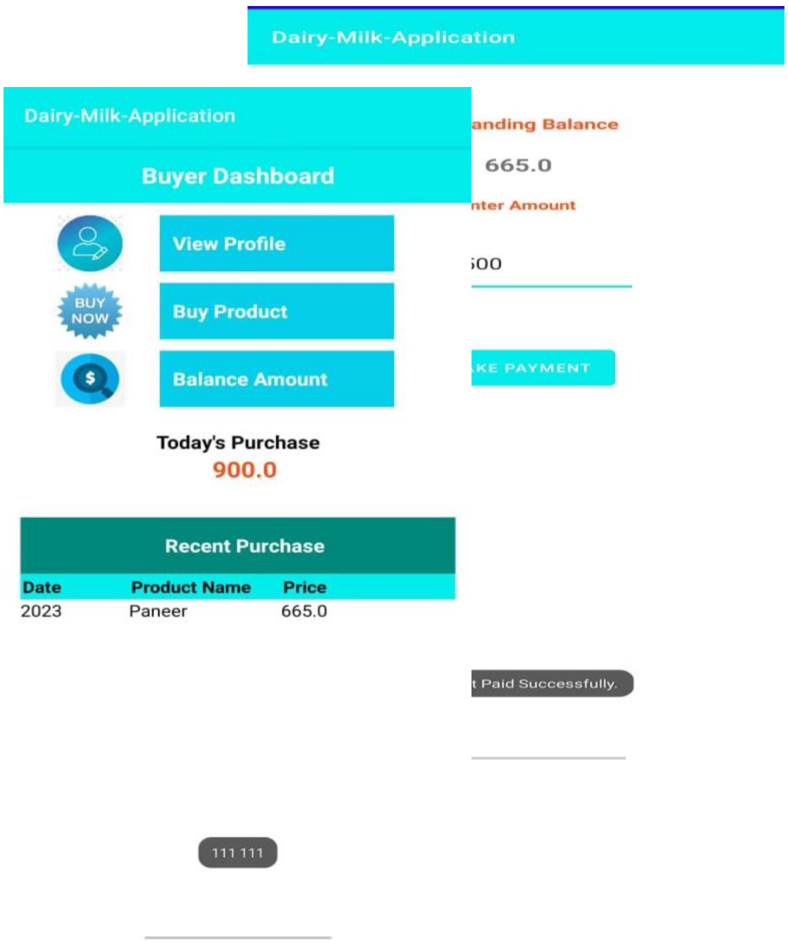
Successfully purchase the product with BALANCE AMOUNT : 235.0 .


Dairy-Milk-Application

Register

Name :	Santosh Hiwale
Dairy Name :	Santosh Dairy
Address :	Vardha
Mobile No. :	64257864
User ID :	123
Password :	•••

REGISTER


Dairy-Milk-Application

Dairy-Milk-Application

Buyer Dashboard

Today's Purchase
900.0

Recent Purchase

Date	Product Name	Price
2023	Paneer	665.0

111 111

anding Balance
665.0
nter Amount
;00

t Paid Successfully.

Chapter 8
Future Expansion

Future Expansion:

Doodh dairy system future work may include:

1. The system can be extended, as the software is constantly evolving and always has a scope for future enhancement.
2. This system is flexible and chances if any change can be made without much difficulty.

The project has a very vast scope in a future project can be updated in near future as and when requirement for the same arises, as it is very flexible in terms of expansion.

The following are the future scope for the project:

- The future scope in this application are adding a chat box so that the end user can use the application in more efficient way.
- A proper payment options will be provided in future for example the online payment.
- Less use of data base and conditioning algorithms and more use of Artificial Intelligence.
- This is a Desktop base application which can also be developed as an Android and IOS application.

Chapter 9

Conclusion

In conclusion we have designed and developed user friendly an Android application, 'Doodh Dairy' for the dairy owners as well as their customers.This app allow the Dairy owners to See the buyers data ,in which customers list is mentioned and the product they ordered and also shows the outstanding amount .It also allow buyers to register themselves and Buy product from the dairy and the daily Milk ,dahi data etc.

References

[1] Ghandi, Li., Catarina S., Martínez, D. and Gualotuña T. (2017) , ".Mobile application development process: A practical experience." 12th Iberian Conference on Information Systems and Technologies (CISTI), 1-6.

[2] Xhafa, Fatos, Santi Caballé, Isaac Rustarazo, and Leonard Barolli.Implementing a mobile campus Using MLE Moodle.In 2010 International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, pp. 207-214. IEEE, 2010.

[3] Biqing, Li, Wenya Lai, Yang, C. and Zheng. S.(2016), "The Design and Implementation of the APP of Experiencing Guangxi Folk Custom."International Conference on Economics and Management Innovations. Atlantis Press, 2016.

[4] Chou, Te-Lien, and Lih-Juan Chan Lin. (2012), "Augmented reality smartphone environment orientation application: a case study of the Fu-Jen University mobile campus touring system." Procedia-Social and Behavioral Sciences 46 : 410-416