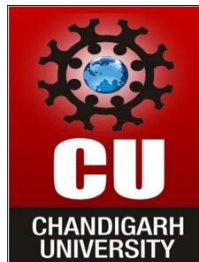


# ***FINAL REPORT***

**BACHELOR OF  
ENGINEERING**

**COMPUTER SCIENCE &  
ENGINEERING**



**UID:  
20BET1072**

**Submitted By:  
Ritik Panwar**

**DEPARTMENT OF COMPUTER  
SCIENCE & ENGINEERING  
CHANDIGARH UNIVERSITY,  
GHARUAN**



effervescence  
IIIT Allahabad



# CERTIFICATE

OF COMPLETION



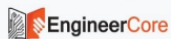
**ritik**

Is presented with this certificate to acknowledge their active participation in

## Android Application Development

Course with **Engineer Core Winter Training** in association with **Effervescence IIIT Allahabad**  
in the month of **January** and **February 2024**

presented by



EN-SI-7949

Certificate Number

  
Mr. Aashish Kapoor  
Program  
Coordinator

  
Mr. Gautam Kumar  
Founder  
Engineer Core

Winter  
Training  
2024

## **ACKNOWLEDGEMENT**

The project work in this report is an outcome of continuous work over a period and drew intellectual support from various sources. I would like to articulate our profound gratitude and indebtedness to those persons who helped us in completion of the project. I take this opportunity to express my sincere thanks and deep gratitude to all those people who extended their wholehearted co-operation and have helped me in completing this project successfully.

I am thankful to iMax Infotech Training Associates for teaching and assisting me in making the project successful. I would also like to thank our parents & other fellow mates for guiding and encouraging me throughout the duration of the project.

## **DECLARATION**

I hereby declare that this project entitled HealthyMe based on Android Application Development submitted by our team is a record of bonafide project work completed during summer training. I further declare that the work reported in this project has not been submitted anywhere else and is not copied from anywhere.

## Table of Contents

1. Introduction to Mobile Technologies.....	01
1.1 Background about Mobile Technologies	
1.2 Android	
1.3 Mobile Application	
2. Introduction to Android.....	03
2.1 Android Architecture	
3. Android Building Blocks.....	06
4. Introduction to the Development tool “Android Studio”.....	10
5. Android UI Design.....	11
6. Database SQLite.....	14
7. Software Requirement Specification.....	17
7.1 Introduction	
7.2 Overall Description	
7.3 Specific Requirements	
7.4 Change Management Process	
7.5 Database Schema	
8. Design.....	21
9. Conclusion.....	23

# 1. Introduction to Mobile Technologies

## 1.1 Background about Mobile Technologies

Mobile technology is the technology used for cellular communication. Since the start of this millennium, a standard mobile device has gone from being no more than a simple two-way pager to being a mobile phone, GPS navigation device, an embedded web browser and instant messaging client, and a handheld game console. Many types of mobile operating systems are available for smart phones, including Android, BlackBerry OS, iOS, Symbian, Windows Mobile and Bada.

## 1.2 Android

Android is an operating system based on Linux with a Java programming interface. Android is a mobile operating system (OS) developed by Google. Android is the first completely open source mobile OS. Building on the contributions of the open-source Linux community and more than 300 hardware, software, and carrier partners, Android has rapidly become the fastest-growing mobile OS.

### Android versions

Code name	Version number	Initial release date	API level	Support status
N/A	1.0	23 September 2008	1	Discontinued
	1.1	9 February 2009	2	Discontinued
Cupcake	1.5	27 April 2009	3	Discontinued
Donut	1.6	15 September 2009	4	Discontinued
Eclair	2.0 – 2.1	26 October 2009	5–7	Discontinued
Froyo	2.2 – 2.2.3	20 May 2010	8	Discontinued

Gingerbread	2.3 – 2.3.7	6 December 2010	9–10	Discontinued
Honeycomb	3.0 – 3.2.6	22 February 2011	11–13	Discontinued
Ice Cream Sandwich	4.0 – 4.0.4	18 October 2011	14–15	Discontinued
Jelly Bean	4.1 – 4.3.1	9 July 2012	16–18	Discontinued
KitKat	4.4 – 4.4.4	31 October 2013	19–20	Security updates only
Lollipop	5.0 – 5.1.1	12 November 2014	21–22	Supported
Marshmallow	6.0 – 6.0.1	5 October 2015	23	Supported
Nougat	7.0 – 7.1	22 August 2016	24–25	Supported

### 1.3 Mobile Application

A mobile application (or mobile app) is a software application designed to run on smart phones, tablet computers and other mobile devices. Mobile apps were originally offered for general productivity and information retrieval, including email, calendar, contacts, and stock market and weather information.

## 2. Introduction to Android

### 2.1 Android Architecture

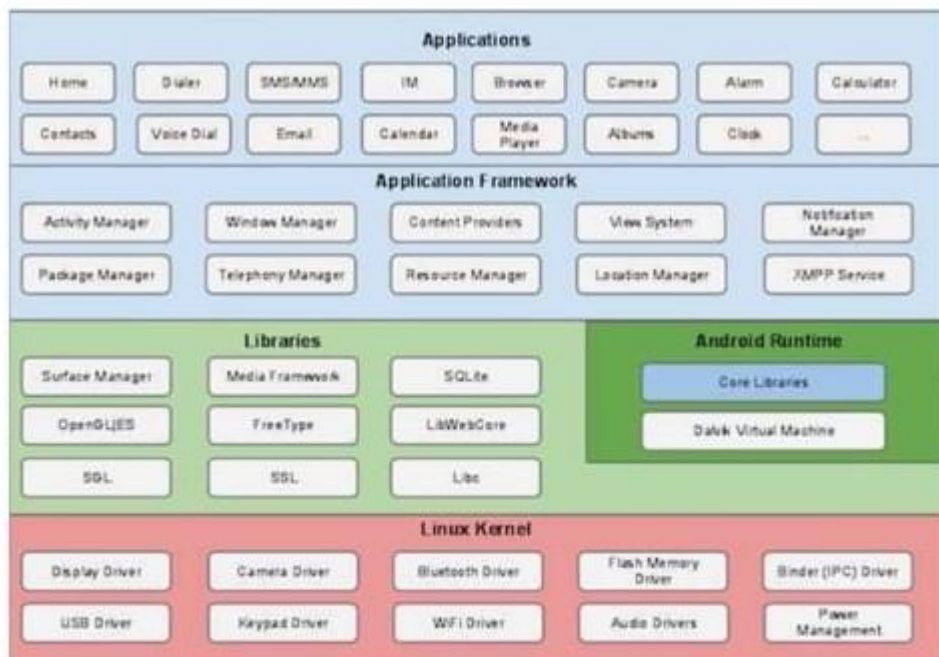
Android operating system is a stack of software components which is roughly divided into five sections and four main layers as shown below in the architecture diagram.

#### Linux kernel

At the bottom of the layers is Linux - Linux 3.6 with approximately 115 patches. This provides a level of abstraction between the device hardware and it contains all the essential hardware drivers like camera, keypad, display etc. Also, the kernel handles all the things that Linux is really good at such as networking and a vast array of device drivers, which take the pain out of interfacing to peripheral hardware.

#### Libraries

On top of Linux kernel there is a set of libraries including open-source Web browser engine WebKit, well known library libe, SQLite database which is a useful repository for storage and sharing of application data, libraries to play and record audio and video, SSL libraries responsible for Internet security etc.





## Android Libraries

This category encompasses those Java-based libraries that are specific to Android development. Examples of libraries in this category include the application framework libraries in addition to those that facilitate user interface building, graphics drawing and database access. A summary of some key core Android libraries available to the Android developer is as follows –

- `android.app` – Provides access to the application model and is the cornerstone of all Android applications.
- `android.content` – Facilitates content access, publishing and messaging between applications and application components.
- `android.database` – Used to access data published by content providers and includes SQLite database management classes.
- `android.opengl` – A Java interface to the OpenGL ES 3D graphics rendering API.
- `android.os` – Provides applications with access to standard operating system services including messages, system services and inter-process communication.
- `android.text` – Used to render and manipulate text on a device display.
- `android.view` – The fundamental building block of application user interfaces.
- `android.widget` – A rich collection of pre-built user interface components such as buttons, labels, list views, layout managers, radio buttons etc.
- `android.webkit` – A set of classes intended to allow web-browsing capabilities to be built into applications.

Having covered the Java-based core libraries in the Android runtime, it is now time to turn our attention to the C/C++ based libraries contained in this layer of the Android software stack.

## Android Runtime

This is the third section of the architecture and available on the second layer from the bottom. This section provides a key component called Dalvik Virtual Machine which is a kind of Java Virtual Machine specially designed and optimized for Android. The Dalvik VM makes use of Linux core features like memory management and multi-threading, which is intrinsic in the Java language. The Dalvik VM enables every Android application to run in its own process, with its own instance of the Dalvik virtual machine. The Android runtime also provides a set of core libraries which enable Android application developers to write Android applications using standard Java programming language

## **Application Framework**

The Application Framework layer provides many higher-level services to applications in the form of Java classes. Application developers are allowed to make use of these services in their applications.

The Android framework includes the following key services –

- Activity Manager - Controls all aspects of the application lifecycle and activity stack.
- Content Providers - Allows applications to publish and share data with other applications.
- Resource Manager - Provides access to non-code embedded resources such as strings, color settings and user interface layouts.
- Notifications Manager - Allows applications to display alerts and notifications to the user.
- View System - An extensible set of views used to create application user interfaces.

## **Applications**

You will find all the Android application at the top layer. You will write your application to be installed on this layer only. Examples of such applications are Contacts Books, Browser, and Games etc.

### 3. Android Building Blocks

Each building block is a different point through which the system can enter your application. Not all components are actual entry points for the user and some depend on each other, but each one exists as its own entity and plays a specific role—each one is a unique building block that helps define your application's overall behavior.

#### 3.1 Activities

An activity is an application component that provides a screen with which users can interact in order to do something, such as dial the phone, take a photo, send an email, or view a map. Each activity is given a window in which to draw its user interface. The window typically fills the screen, but may be smaller than the screen and float on top of other windows. An application usually consists of multiple activities that are loosely bound to each other. Typically, one activity in an application is specified as the "main" activity, which is presented to the user when launching the application for the first time. Each activity can then start another activity in order to perform different actions. Each time a new activity starts, the previous activity is stopped, but the system preserves the activity in a stack (the "back stack"). When a new activity starts, it is pushed onto the back stack and takes user focus. The back stack abides to the basic "last in, first out" stack mechanism, so, when the user is done with the current activity and presses the *Back* button, it is popped from the stack (and destroyed) and the previous activity resumes. When an activity is stopped because a new activity starts, it is notified of this change in state through the activity's lifecycle callback methods. There are several callback methods that an activity might receive, due to a change in its state—whether the system is creating it, stopping it, resuming it, or destroying it—and each callback provides you the opportunity to perform specific work that's appropriate to that state change. For instance, when stopped, your activity should release any large objects, such as network or database connections. When the activity resumes, you can reacquire the necessary resources and resume actions that were interrupted. These state transitions are all part of the activity lifecycle.



Fig. 1 Transition in between activities

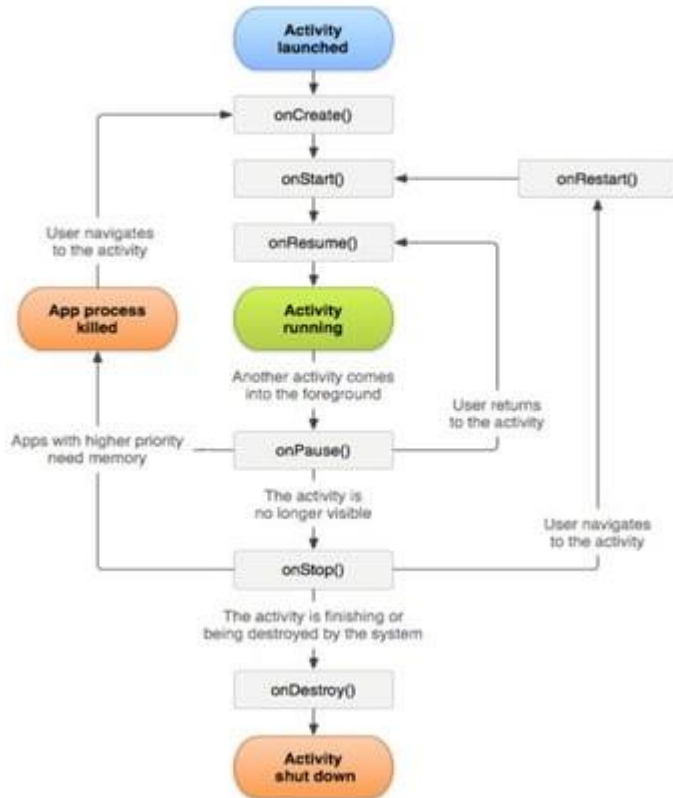


Fig. 2 Lifecycle of an activity

## 3.2 Services

A service is an application component that can perform long-running operations in the background, and it does not provide a user interface. Another application component can start a service, and it continues to run in the background even if the user switches to another application. Additionally, a component can bind to a service to interact with it and even perform inter process communication (IPC). For example, a service can handle network transactions, play music, perform file I/O, or interact with a content provider, all from the background.

These are the three different types of services:

1. Scheduled

A service is scheduled when an API such as the `JobScheduler`, introduced in Android 5.0 (API level 21), launches the service. You can use the `JobScheduler` by registering jobs and specifying their requirements for network and

timing. The system then gracefully schedules the jobs for execution at the appropriate times. The `JobScheduler` provides many methods to define service-execution conditions.

## 2. Started

A service is started when an application component (such as an activity) calls `startService()`. After it's started, a service can run in the background indefinitely, even if the component that started it is destroyed. Usually, a started service performs a single operation and does not return a result to the caller. For example, it can download or upload a file over the network. When the operation is complete, the service should stop itself.

## 3. Bound

A service is bound when an application component binds to it by calling `bindService()`. A bound service offers a client-server interface that allows components to interact with the service, send requests, receive results, and even do so across processes with inter process communication (IPC). A bound service runs only as long as another application component is bound to it. Multiple components can bind to the service at once, but when all of them unbind, the service is destroyed.

### 3.3 Content Providers

A content provider manages a shared set of application data. You can store the data in the file system, an SQLite database, on the web, or any other persistent storage location your application can access. A content provider provides a structured interface to application data. Via a content provider your application can share data with other applications. Android contains an SQLite database which is frequently used in conjunction with a content provider.

### 3.4 Broadcast Receivers

Broadcast Receivers simply respond to broadcast messages from other applications or from the system itself. These messages are sometime called events or intents. For example, applications can also initiate broadcasts to let other applications know that some data has been downloaded to the device and is available for them to use, so this is broadcast receiver who will intercept this communication and will initiate appropriate action.

### 3.5 Intent

Intents are asynchronous messages which allow the application to request functionality from other Android components, e.g. from services or activities. An application can call a component directly (explicit Intent) or ask the Android system to evaluate registered components based on the intent data (implicit intents). For example the application could implement sharing of data via an intent and all components which allow sharing of data would be available for the user to select. Applications register themselves to an intent via an intent. Filter Intents allow an Android application to start and to interact with components from other Android applications.

### 3.6 Using intent to launch the activities

There are separate methods for activating each type of component:

- You can start an activity (or give it something new to do) by passing Intent to `startActivity()` or `startActivityForResult()` .
- You can start a service (or give new instructions to an ongoing service) by passing an Intent to `startService()`. Or you can bind to the service by passing an Intent to `bindService()`.
- You can initiate a broadcast by passing an Intent to methods like `sendBroadcast()`, `sendOrderedBroadcast()`, or `sendStickyBroadcast()`.
- You can perform a query to a content provider by calling `query` on a `ContentResolver`.

## 4. Introduction to the Development tool “Android Studio”

Android Studio is the official integrated development environment (IDE) for Android platform development. Android Studio is freely available under the apache license. Android Studio is designed specifically for Android development.

Android application development can be started on either of the following operating systems –

- Microsoft® Windows® 8/7/Vista/2003 (32 or 64-bit).
- Mac® OS X® 10.8.5 or higher, up to 10.9 (Mavericks).
- GNOME or KDE desktop.

All the required tools to develop Android applications are open source and can be downloaded from the Web. Following is the list of softwares that is needed before starting Android application programming.

- Java JDK5 or later version
- Java Runtime Environment (JRE) 6
- Android Studio

## 5. Android UI Design

### 5.1 Introducing Layouts

#### Frame Layout

Frame layouts are one of the simplest layout types used to organize controls within the user interface of an Android application. They are used less often than some other layouts, simply because they are generally used to display only one view, or views which overlap. The efficiency of a frame layout makes it a good choice for screens containing few view controls (home screens, game screens with a single canvas, and the like). Sometimes other inefficient layout designs can be reduced to a frame layout design that is more efficient, while other times a more specialized layout type is appropriate. Frame layouts are the normal layout of choice when you want to overlap views.

#### Linear Layout

Linear layouts are one of the simplest and most common types of layouts used by Android developers to organize controls within their user interfaces. The linear layout works much as its name implies: it organizes controls linearly in either a vertical or horizontal fashion. When the layout's orientation is set to vertical, all child controls within it are organized in a single column; when the layout's orientation is set to horizontal, all child controls within it are organized in a single row. Some of the most important attributes you'll use with linear layouts include:

- The orientation attribute (required), which can be set to vertical or horizontal.
- The gravity attribute (optional), which controls how all child controls are aligned and displayed within the linear layout (class: `LinearLayout`).
- The layout\_weight attribute (optional, applied to each child control) specifies each child control's relative importance within the parent linear layout (class: `LinearLayout.LayoutParams`).

#### Relative Layout

The relative layout works much as its name implies: it organizes controls relative to one another, or to the parent control itself. It means that child controls, such as `ImageView`, `TextView`, and `Button` controls, can be placed above, below, to the left or right, of one another. Child controls can also be placed in relation to the parent (the relative layout container); including placement of controls aligned to the top, bottom, left or right edges of the layout.

Some specific attributes apply to relative layouts—namely the child rules, including:

- Rules for child control centering within the parent layout, including: center horizontally, center vertically, or both.
- Rules for child control alignment within the parent layout, including: align with top, bottom, left or right edge of another control.



- Rules for child control alignment in relation to other child controls, including: align with top, bottom, left or right edge.
- Rules for child control placement in relation to other child controls, including: placement to the left or right of a specific control, or above or below another control.

## **Table Layout**

A table layout is exactly what you might expect: a grid of made up of rows and columns, where a cell can display a view control. From a user interface design perspective, a `TableLayout` is comprised of `TableRow` controls—one for each row in your table. The contents of a `TableRow` are simply the view controls that will go in each “cell” of the table grid. Although table layouts can be used to design entire user interfaces, they usually aren’t the best tool for doing so, as they are derived from `LinearLayout` and not the most efficient of layout controls. However, for data that is already in a format suitable for a table, such as spreadsheet data, table layout may be a reasonable choice.

## **5.2 Different UI widgets available in Android**

- Text View
- Edit Text
- List View
- Spinner
- Button
- Check Box
- Radio Button
- Scroll View

## **5.3 Menus**

### **Options menu and action bar**

The options menu is the primary collection of menu items for an activity. It’s where you should place actions that have a global impact on the app, such as “Search”, “Compose email”, and “Settings”. On Android 3.0 and higher, items from the options menu are presented by the action bar as a combination of on-screen action items and overflow options. Beginning with Android 3.0, the `Menu` button is deprecated (some devices don’t have one), so you should migrate toward using the action bar to provide access to actions and other options.

### **Context Menu and Contextual Action Mode**

A context menu is a floating menu that appears when the user performs a long-click on an element. It provides actions that affect the selected content or context frame. A contextual menu offers actions that affect a specific item

or context frame in the UI. You can provide a context menu for any view, but they are most often used for items in a `ListView`, `GridView`, or other view collections in which the user can perform direct actions on each item.

## **Popup Menu**

A popup menu displays a list of items in a vertical list that's anchored to the view that invoked the menu. It's good for providing an overflow of actions that relate to specific content or to provide options for a second part of a command.

## **5.4 Dialog Box**

A dialog is a small window that prompts the user to make a decision or enter additional information. A dialog does not fill the screen and is normally used for modal events that require users to take an action before they can proceed.

## **5.5 Toast**

A toast provides simple feedback about an operation in a small popup. It only fills the amount of space required for the message and the current activity remains visible and interactive.

## **5.6 Adapters**

An Adapter acts as a bridge between a `ListView` and the underlying data for that view. The Adapter provides access to the data items. The Adapter is also responsible for making a View for each item in the data set.

### **Array Adapter**

Array Adapter is a concrete `BaseAdapter` that is backed by an array of arbitrary objects. By default this class expects that the provided resource id references a single `TextView`. If you want to use a more complex layout, use the constructor that also takes a field id. That field id should reference a `TextView` in the larger layout resource.

## **5.7 Notification Manager**

A notification is a message you can display to the user outside of your application's normal UI. When you tell the system to issue a notification, it first appears as an icon in the notification area. To see the details of the notification, the user opens the notification drawer. Notification Manager Class is used to notify the user of events that happen. This is how you tell the user that something has happened in the background. Notifications can take different forms:

- A persistent icon that goes in the status bar and is accessible through the launcher. (when the user selects it, a designated Intent can be launched).
- Turning on or flashing LEDs on the device.
- Alerting the user by flashing the backlight, playing a sound, or vibrating.

## 6. Database SQLite

SQLite is an Open Source database. SQLite supports standard relational database features like SQL syntax, transactions and prepared statements. The database requires limited memory at runtime (approx. 250 Kbytes) which makes it a good candidate from being embedded into other runtimes.

SQLite is embedded into every Android device. Using a SQLite database in Android does not require a setup procedure or administration of the database. We only have to define the SQL statements for creating and updating the database. Afterwards the database is automatically managed for us by the Android platform. If our application creates a database, this database is by default saved at directory - DATA/data /APP\_NAME/ databases /FILENAME.

SQLiteDatabase is the base class for working with a SQLite database in Android and provides methods to open, query, update and close the database.

- SQLiteDatabase provides the insert(), update() and delete() methods.
- The object ContentValues allows to define key/values. The key represents the table column identifier and the value represents the content for the table record in this column. ContentValues can be used for inserts and updates of database entries.
- Queries can be created via the.rawQuery() and query() methods or via the SQLiteQueryBuilder class .
- .rawQuery() directly accepts an SQL select statement as input.
- query() provides a structured interface for specifying the SQL query.
- SQLiteQueryBuilder is a convenience class that helps to build SQL queries.

### 6.1 SQLiteOpenHelper

To create and upgrade a database in our Android application we create a subclass of the SQLiteOpenHelper class. In the constructor of our subclass we call the super() method of SQLiteOpenHelper, specifying the database name and the current database version. In this class we need to override the following methods to create and update our database-

- onCreate() is called by the framework, if the database is accessed but not yet created.
- onUpgrade() called, if the database version is increased in your application code. This method allows you to update an existing database schema or to drop the existing database and recreate it via the onCreate() method.

## 6.2 Query()

The following gives an example of a query() call-

```
return database.query(DATABASE_TABLE, new String[] { KEY_ROWID, KEY_CATEGORY, KEY_SUMMARY, KEY_DESCRIPTION }, null, null, null, null, null);
```

## 6.3 Opening and Closing a Database

```
SQLiteDatabase db = this.getWritableDatabase(); //Opening DatabaseConnection
```

```
ContentValues values = new ContentValues();
```

```
values.put(KEY_NAME, contact.getName()); //Contact Name
```

```
values.put(KEY_PH_NO, contact.getPhoneNumber()); //Contact Phone Number
```

```
db.insert(TABLE_CONTACTS, null, values); //Inserting Row
```

```
db.close(); //Closing database connection
```

## 6.4 Cursor

Cursor provides typed get\*() methods, e.g. getLong(columnIndex), getString(columnIndex) to access the column data for the current position of the result. The "columnIndex" is the number of the column you are accessing. A Cursor needs to be closed with the close() method call.

### Insert

```
ContentValues values = new ContentValues();
```

```
values.put(MySQLiteHelper.COLUMN_COMMENT, comment);
```

```
long insertId = database.insert(MySQLiteHelper.TABLE_COMMENTS, null, values);
```

```
cursor.close();
```

```
return newComment;
```

### Delete

```
database.delete(MySQLiteHelper.TABLE_COMMENTS, MySQLiteHelper.COLUMN_ID + " = " + id, null);
```

## 6.5 Content provider and Sharing data

A SQLite database is private to the application which creates it. If you want to share data with other applications you can use a content provider. A content provider allows applications to access data. In most cases this data is stored in an SQLite database. While a content provider can be used within an application to access data, its is typically used to share data with other application. As application data is by default private, a content provider is a convenient to share your data with other application based on a structured interface. A content provider must be declared in the `AndroidManifest.xml` file.

## **7. Software Requirement Specification**

### **7.1 Introduction**

#### **Purpose**

HealthyMe is a fitness android application that aims at providing a complete guide to be fit and healthy. It comprehensively covers details of various aspects of health for different age groups. This document provides all the required specifications of the android application.

#### **Scope**

The application contains following in-built features-

- Set up alarms for various activities like exercise, morning walk, etc.
- BMI calculator
- Issues related to mental and physical health.

#### **Abbreviations**

API: Application Program Interface

RAM: Random Access Memory

GUI: Graphical User Interface

SRS: Software Requirement Specification

DB: Database

IDE: Integrated Development Environment

#### **Overview of SRS**

The SRS aims at providing all the internal and external details of the application HealthyMe. The next section gives the overall description of the application, some assumptions taken, features provided, etc. The third section provides the specific requirements of the software including functional and non-functional requirements. Section 4 includes the add-ons which can be given to the application in near future.

## 7.2 Overall Description

### Product Perspective

Software interface: This application will be made using the IDE Android Studio with Java and XML. It shall use SQLite at the database level to store details. It shall require Android API 17.

Memory Constraints: The application shall run on a system with RAM>2GB.

Communication interface: The user shall interact with the application with the help of graphical screens. The user can select the criteria according to their age and can use various applications such as alarm, BMI calculator, diet etc.

### Product Functions

The application contains following in-built features-

- The user shall sign-up with the details when the application is downloaded for the first time and remains logged in thereafter.
- The user shall choose the age criteria to which he belongs can use the mentioned services.
- On the basis of the selection of service the results will be shown accordingly.
- If the user is registered his information details will be stored in database forever and he can login for the next time.

### User Characteristics

The user should know how to operate a smart phone with android OS and he should be familiar with the functioning of android applications. He should know how to input data and select options.

### Constraints

- It is single-user application.
- The user can expect more features or want to gather more information.

## 7.3 Specific Requirements

### External Interface

The application connects to an external SQLite database stored in the software.

### Functional Requirements

Use-case

Primary Actor: User

Pre-conditions: Application should be installed successfully.

Main scenario:

1. The user shall choose the age criteria to which he belongs can use the mentioned services.
2. On the basis of the selection of service, the results will be shown accordingly.
3. If the user is registered his information details will be stored in database forever and he can login for the next time.

Alternate scenario:

1. The application is not installed.
2. User doesn't know how to operate smart phones.

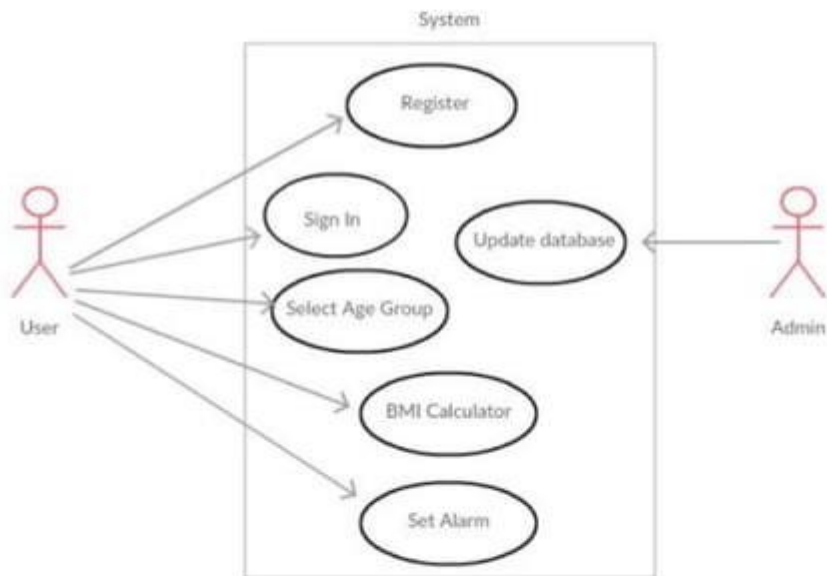


Fig. 3 Use-Case diagram of the application

## Performance Requirement

The performance and efficiency depends upon the RAM of the mobile.



## Logical Database Requirements

The android app shall use the database made using MySQL 5.5. The database will consist of tables containing user details.

## Software System Attributes

- Portability- The application is made in java so it is portable on all android versions above JellyBean.
- Reliability- The android application shall provide reliable and authenticated information.
- Security- The user data shall be stored securely in the database.

## 7.4 Change Management Process

Another extension of the application can be adding a GPS tracker which will keep a count of distance walked by the user every day.

## 7.5 Supporting Information/Database Schema

SQLite is used for database. The following table is made to store the credentials of the users-

### Register table

Name	Email	Password	Age-criteria	Gender
------	-------	----------	--------------	--------

## 8. Design

Software Design encompasses the set of principles, concepts and practices that led to the development of a high quality system or product. The goal of design is to produce a model that exhibits firmness, commodity and delight. Once the requirements have been analyzed and modelled, software design sets the stage for construction of the software. Each of the elements of the analysis model provides information for a complete specification of design.

A flowchart is a type of diagram that represents an algorithm workflow or process, showing the steps as boxes of various kinds, and their order by connecting them with arrows. This diagrammatic representation illustrates a solution model to a given problem. Flowcharts are used in analyzing, designing, documenting or managing a process or program in various fields.

- A processing step, usually called activity, is denoted as a rectangular box.
- A decision is usually denoted as a diamond.

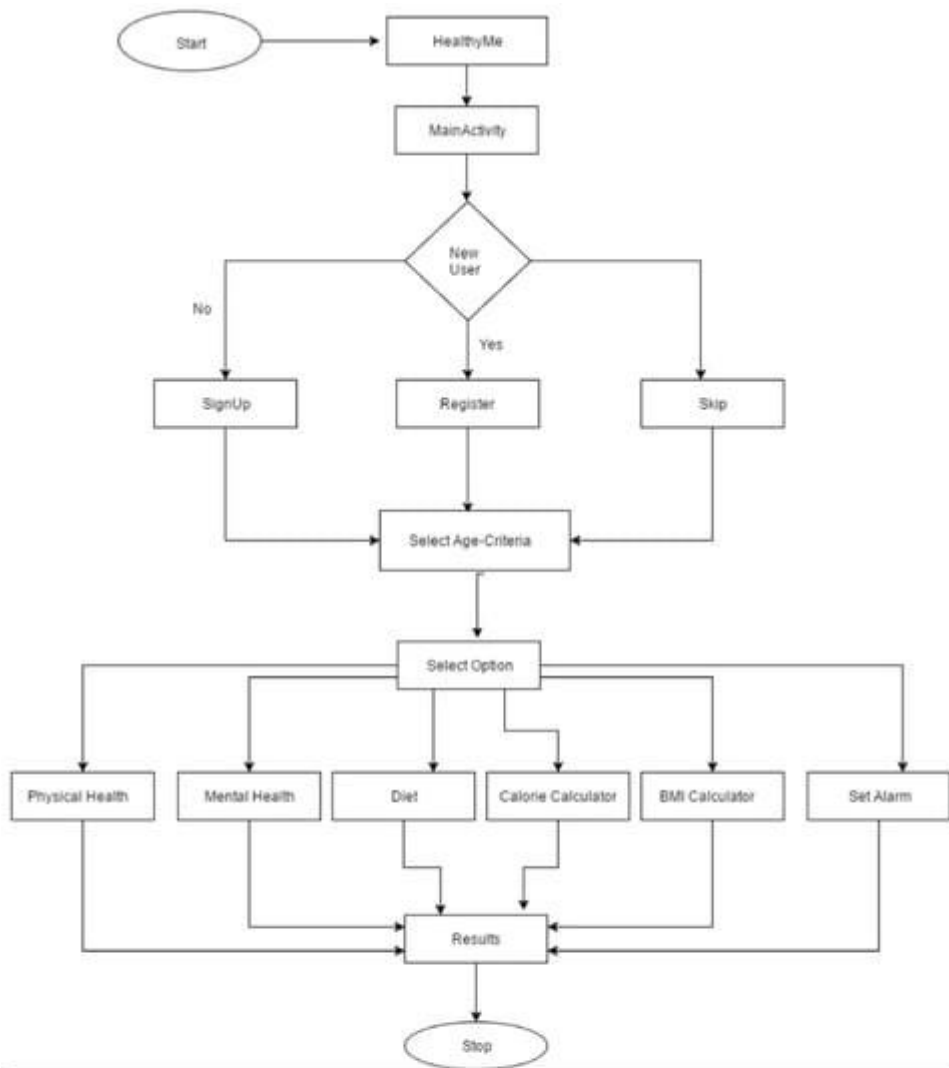



Fig. 4 Flow chart Representation of the application

Some of the activities are shown below-

- **Register Activity**



**Features and Age-Criteria**



- **Set Alarm activity**



## 9. Conclusion

Android Smartphone are in hype in the 21<sup>st</sup> century. The scope of android applications is increasing day by day. Its development has become an essential part of today's programming curriculum. The project HealthyMe is an android app that incorporates everything from UI design to database. Its utility and efficiency is also very high. More such android apps can be developed using similar concepts and tools. The society has a dearth of ideas. These ideas can be most effectively implemented by developing user-friendly android applications. Through this project, we got to learn a lot, including, database connectivity using SQLite. Being new to app development, we came to know a lot about developing an android application from scratch.

## References

- 1) <https://developer.android.com>
- 2) *Android Architecture* Retrieved from <https://tutorialspoint.com>
- 3) [stackoverflow.com](https://stackoverflow.com)
- 4) [www.quora.com](https://www.quora.com)

