

*A Project Report*

*on*

## **Cipher Dossier**

*Submitted in partial fulfillment of the requirements*

*for the award of the degree of*

## **BACHELOR OF TECHNOLOGY**

*in*

## **Computer Science & Engineering**

*by*

I. Sai Sridivya	(164G1A0590)
S. Sajida	(164G1A0592)
D. Skandha	(164G1A0599)
D. Sreeveni	(164G1A05A2)

**Under the Guidance of**

**G. Hemanth Kumar Yadav**, M.Tech(Ph.D)  
**Assistant Professor**



**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**(B.Tech Program Accredited by NBA)**

**SRINIVASA RAMANUJAN INSTITUTE OF TECHNOLOGY: ANANTAPURAMU**

(Accredited by NAAC with 'A' Grade, Affiliated to JNTUA, Approved by AICTE, New Delhi)

**2019-2020**



## Certificate

This is to certify that the project report entitled **Cipher Dossier** is the bonafide work carried out by **I. Sai Sridivya** bearing Roll Number **164G1A0590**, **S. Sajida** bearing Roll Number **164G1A0592**, **D. Skandha** bearing Roll Number **164G1A0599** and **D. Sreeveni** bearing Roll Number **164G1A05A2** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science & Engineering** during the academic year 2019-2020.

**Guide**

Mr.G. Hemanth Kumar Yadav,M.Tech(Ph.D)  
Assistant Professor

**Head of the Department**

Dr. G.K.V. Narasimha Reddy,Ph.D  
Professor & HOD

Date:

**EXTERNAL EXAMINER**

Ananthapuramu

## ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of people who made it possible, whose constant guidance and encouragement crowned our efforts with success. It is a pleasant aspect that we have now the opportunity to express my gratitude for all of them.

It is with immense pleasure that we would like to express my indebted gratitude to my Guide **Mr.G.Hemanth Kumar Yadav,M.Tech(Ph.D),Computer Science & Engineering**,who has guided me a lot and encouraged me in every step of the project work. We thank him for the stimulating guidance, constant encouragement and constructive criticism which have made possible to bring out this project work.

We express our deep-felt gratitude to **Mr.R.Sandeep Kumar,M.Tech(Ph.D), Assistant Professor**, project coordinator valuable guidance and unstinting encouragement enable us to accomplish our project successfully in time.

We are very much thankful to **Dr. G.K.V.Narasimha Reddy, Ph.D, Professor & Head of the Department, Computer Science & Engineering**, for his kind support and for providing necessary facilities to carry out the work.

We wish to convey my special thanks to **Dr.T.Hitendra Sarma,Ph.D, Principal of Srinivasa Ramanujan Institute of Technology** for giving the required information in doing our project work. Not to forget, we thank all other faculty and non-teaching staff, and my friends who had directly or indirectly helped and supported us in completing our project in time.

We also express our sincere thanks to the Management for providing excellent facilities.

Finally, we wish to convey our gratitude to our family who fostered all the requirements and facilities that we need.

**Project Associates**

## **DECLARATION**

We I. Sai Sridivya bearing reg no : 164G1A0590, S. Sajida bearing reg no : 164G1A0592, D. Skandha bearing reg no : 164G1A0599, D. Sreeveni bearing reg no : 164G1A05A2 students of SRINIVASA RAMANUJAN INSTITUTE OF TECHNOLOGY, Rotarypuram, hereby declare that the dissertation entitled “CIPHER DOSSIER” embodies the report of our project work carried out by us during IV Year Bachelor of Technology under the guidance of Mr.G. Hemanth Kumar Yadav,M.Tech(Ph.D), Department of CSE and this work has been submitted for the partial fulfillment of the requirements for the award of Bachelor of Technology degree.

The results embodied in this project report have not been submitted to any other Universities of Institute for the award of Degree.

I. Sai Sridivya	Reg no: 164G1A0590
S. Sajida	Reg no: 164G1A0592
D. Skandha	Reg no: 164G1A0599
D. Sreeveni	Reg no: 164G1A05A2

# Contents

<b>List of Figures</b>	viii
<b>List of Abbreviations</b>	xi
<b>List of Tables</b>	xii
<b>List of Output Screens</b>	xiii
<b>Abstract</b>	xiv
<b>Chapter 1 : Introduction</b>	1
1.1 Motivation	1
1.2 Problem Definition	1
1.3 Objective of Project	2
1.4 Scope of the Project	2
<b>Chapter 2 : Literature Survey</b>	3
2.1 Introduction	3
2.2 Existing System	3
2.2.1 CryptAll	3
2.3 Disadvantages of Existing System	4
2.4 Proposed System	4
2.4.1 Cipher Dossier	4
2.5 Summary	4
<b>Chapter 3 : Analysis</b>	5
3.1 Introduction	5
3.2 SDLC Model	6
3.3 Software Requirement Specification	8
3.3.1 User Requirements	9
3.3.2 Software Requirements	9
3.3.3 Hardware Requirements	10
3.4 Android Installation Procedure	10
3.5 Summary	18

<b>Chapter 4 : Design</b>	19
4.1 Introduction	19
4.2 UML Diagrams	19
4.2.1 Use Case Diagram	20
4.2.2 Sequence Diagram	21
4.2.3 Class Diagram	23
4.2.4 State Chart Diagram	23
4.2.5 Activity Diagram	24
4.2.6 Deployment Diagram	25
4.2.7 Component Diagram	26
4.2.8 Collaboration Diagram	26
4.3 Module Design and Organization	27
4.4 Summary	27
<b>Chapter 5 : Implementation &amp; Results</b>	28
5.1 Introduction	28
5.2 Explanation of Key Functions	28
5.2.1 Firebase	28
5.2.1.1 Services of Firebase	29
5.2.2 Room Database	30
5.2.3 RecyclerView	31
5.2.4 AES Algorithm	32
5.3 Method of Implementation	33
5.3.1 Output Screens	33
5.4 Summary	42
<b>Chapter 6 : Testing &amp; Validation</b>	43
6.1 Introduction	43
6.2 Design of Test cases and scenarios	43
6.2.1 Unit Testing	43
6.2.2 Integration Testing	45
6.2.3 Espresso Testing	45

6.3	Validation of Test cases	48
6.4	Summary	59
<b>Conclusion</b>		50
<b>Bibliography</b>		51

## **List of Figures**

<b>Fig No</b>	<b>Description</b>	<b>Page No</b>
3.1	Agile Model	6
3.2	Setup Android Studio	11
3.3	Installation of Android SDK and AVD	11
3.4	Set the Android Studio and Android SDK installation locations	11
3.5	Create a new shortcut for Android Studio	12
3.6	Installation in progress	12
3.7	Leave the Start Android Studio checkbox checked to run this software	13
3.8	Android studio start's screen	13
3.9	Import settings	13
3.10	Validate your Android SDK and development environment setup	14
3.11	Choose an installation step	14
3.12	Review settings	15
3.13	Welcome to Android Studio	15
3.14	Specify an activity template	16
3.15	Customize your activity	16
3.16	Android Studio workspace	17
3.17	The project and editor windows	17
4.1	Use case Diagram	21
4.2	Sequence Diagram	22
4.3	Class Diagram	23
4.4	State Chart Diagram	24
4.5	Activity Diagram	24



4.6	Deployment Diagram	25
4.7	Component Diagram	26
4.8	Collaboration Diagram	27
5.1	Code in XML file for recycler view	32
5.2	Encrypt Method	32
5.3	Decrypt Method	33
5.4	Login Page	33
5.5	Registration Page	34
5.6	Home Activity	34
5.7	Settings Activity	35
5.8	Entering Key	35
5.9	Contacts Activity	36
5.10	After adding Contacts	36
5.11	Messages activity	37
5.12	Notes activity	37
5.13	Adding Notes	38
5.14	Files Activity	38
5.15	Images Activity	39
5.16	Videos Activity	39
5.17	Audio Activity	40
5.18	Documents Activity	40
5.19	Zip files after adding zip file.	41
5.20	Uploading to cloud	41
5.21	Contacts activity in another mobile	42

6.1	Dependencies for executing the espresso testing	45
6.2	Annotations used in testing classes	46
6.3	Steps involved in Espresso testing	47
6.4	Validating the email address	48
6.5	Showing alert Message	48

## **List of Abbreviations**

PDA	Personal Digital Assistants
AES	Advanced Encryption Standard
SDLC	Software Development Life Cycle
UML	Unified Modelling Language
CONOPS	Concept of Operations
IDE	Integrated Development Environment
JDK	Java Development Kit
SDK	Software Development Kit
JDBC	Java Database Connectivity
REST	Representational State Transfer
RMI	Remote Method Invocation
GPS	Global Positioning System
SRS	Software Requirement Specification
EEN	Execution Environment Node
GCM	Google Cloud Messaging
FCM	Firebase Cloud Messaging
API	Application Program Interface
HTTP	Hyper Text Transfer Protocol
CSS	Cascading Style Sheets
HTML	Hyper Text Markup Language
CDN	Content Delivery Network
SSL	Secure Socket Layer

## **List of Tables**

<b>S.No</b>	<b>Description</b>	<b>Page No</b>
6.1	Unit Testing Modules	43

## **List of Output Screens**

<b>S.No</b>	<b>Description</b>	<b>Page No</b>
5.2	Login page	32
5.3	Registration Page	33
5.4	Home Activity	33
5.5	Settings Activity	34
5.6	Entering Key	34
5.7	Contacts Activity	35
5.8	After Adding Contacts	35
5.9	Messages Activity	36
5.10	Notes Activity	36
5.11	Adding Notes	37
5.12	Files Activity	37
5.13	Images Activity	38
5.14	Videos Activity	38
5.15	Audios Activity	39
5.16	Documents Activity	39
5.17	Zip Files Activity	40
5.18	Uploading to cloud	40
5.19	Contacts activity in another mobile	41

## **ABSTRACT**

Security these days has a major role for keeping the data away from intruders who tries to access the data with malicious intent. To prevent these malicious intrusions into the system, the data must be in a form that the intruders are unable to get access to confidential data and even if they get access over the data they must not be able to read or modify the data. For this, the data must be encrypted using some encryption mechanisms which are impossible to be cracked by the intruders .The encryption is done by using keys that are to be distributed between the sender and receiver for data exchange or to encrypt the data in the system. The keys used will be known to the user and also to the server which is made use of as the platform. This is applied to the data both in the laptops as well as mobile phones.

The application mainly concentrates on mobile phones and efficiently handles the situation when a mobile is lost and when there is no way to retrieve the data. In this application, the data like contacts, messages, notes, images and videos can be encrypted using a key generated by the user and this key is used to encrypt the contents that are chosen by the user to keep them confidential. The user has to login into the application to make use of the application. The data in this application is stored on the cloud storage and stored in an encrypted form using the key that is generated by the user. Even if the mobile is lost, no one can access the data as it will be encrypted by the key that is known to the user only. The data is decrypted only by the key that is used to encrypt it and if the key is changed, the data cannot be retrieved in its original form. And if the user wants to access the data he can login into the application with their credentials from any device using the key that is used to encrypt the data.

# CHAPTER-1

## INTRODUCTION

### 1.1 Motivation

In today's world most of the communication is done using electronic media. Data Security plays a vital role in such communication. The pervasive use of wireless networks and mobile devices has been changing our living style significantly. Digital media can be transmitted easily in real time anywhere at any time due to the advanced development of communications, Internet and multimedia technology. Information availability has increased dramatically with the advent of mobile devices. Along with great convenience and efficiency, there are new challenges in protecting sensitive and/or private data in these devices. Statistics show that 22% of PDA owners have lost their devices, and 81% of those lost devices had no protection. Even worse, 37% of PDAs have sensitive information on them, such as bank account information, corporate data, passwords, and more. Hence, there is a need to protect data from malicious attacks. However, with this availability comes a problem of maintaining the security of information in the mobile applications.

Secure storage of information is essential; a strong encryption system is required to protect the data stored in android phones at a maximum level. It is necessary to create means which will enable a user to encrypt confidential data present in the mobile phones and decrypt it to recover the encrypted data.

### 1.2 Problem Definition

There will be important files, notes, contacts and messages that need to be secured from intruders. Also, if the mobile is lost which consists of this data that is important they cannot be retrieved.

As a solution to this problem, a mobile application is created where a user can login into the application and can be able to store all the data which will be encrypted with the key given by the user using AES algorithm [1].

### 1.3 Objective of Project

The objective of this project is to keep the data like contacts, messages, notes, files, files like images, videos, audio, documents, etc., safe from intruders. The user is provided the slots to store his data and he can encrypt those files by a secret key, then he can encrypt it and he can save to drive also.

If mobile is lost he can install that application from play store and login with his credentials. He can retrieve his data.

### 1.4. Scope of the Project

*Cipher Dossier* provides the user access only to a limited amount of the data. Also the user has to remember their credentials to login into the application and also the passwords set, otherwise it can be a problem to access the data in the application.



## **CHAPTER-2**

### **LITERATURE SURVEY**

#### **2.1 Introduction**

Security of data is becoming an important factor for a wide spectrum of applications, including communication systems, wireless devices, and is becoming an escalating concern in an increasingly multimedia defined world. Along with great convenience and efficiency, the progress of technology also brings new challenges in protecting sensitive and/or private information carried in these devices. New vulnerability results from unique characteristics of mobile devices. For instance, due to constraints imposed by limited computing power, storage space, and battery lifetime, a light-weight rather than computing intensive and complex encryption algorithm is desired in the mobile devices. Resistance against known attacks is one of the main properties that an encryption algorithm needs to provide. When a new attack is demonstrated as effective, the update of the encryption system is a real necessity to guarantee the security of data [2].

The most challenging part of mobile device data protection lies in the conflicting requirements for the data encryption scheme. Encryption is the basic means of protection, especially on the theft of data and destructive activities. While it should be computationally infeasible for adversaries to decrypt the data in captured mobile devices, the encryption/decryption operation should be reasonably efficient for legitimate users. Furthermore, the required computations should not consume too much energy so as to minimize battery drain.

#### **2.2 Existing System**

##### **2.2.1 CryptAll**

The proposed Android application named 'CryptAll' is a Data Security Android Mobile Application which uses AES-256 bit Encryption and Decryption Algorithm for encryption and Decryption. CryptAll application allows users to safeguard their data by encrypting the files and securing it. The user will be provided with the list of all the encrypted files. The user can select any of the encrypted files and decrypt it to recover the data [3].

## 2.3 Disadvantages of Existing System

- Information is stored in the mobile phones memory itself.
- Access to the data is not available if the mobile phone is lost.
- Cannot hide media from Gallery.

## 2.4 Proposed System

### 2.4.1 Cipher Dossier

The proposed system is “Cipher Dossier”. The user needs to install the application from play store and register in the android application and with those registered credentials he can access the android application.

The user can store data like contacts, messages, notes, files in this android application by uploading them into this android application. When these files are marked they are relocated to the application's storage and it's removed from their original source. Therefore only authorized users are able to see the files and perform operations on them. He can encrypt the sensitive data by a key which user has to provide using which by the AES algorithm which is one of the encryption techniques [4]. The data will be encrypted using the AES algorithm using the key provided by the user. The encrypted data is stored in the mobile itself.

If the mobile phone is lost, the user can install our mobile application and sign-in with the credentials using which they registered in the android application. By this, he gets to access their data which they have uploaded in the android application using the backup option. Therefore, the sensitive information will not be known by the adversaries who have the mobile. By this, the data will be safe in this android application which the user can access from anywhere using the android application.

## 2.5 Summary

Literature survey helps us to know the functionality about the existing system and provides to create a mobile application on Cipher Dossier with new features that does not exist already.

## CHAPTER-3

### ANALYSIS

#### 3.1 Introduction

Android is a mobile operating system developed by Google , based on a modified version of the Linux kernel and other open source software and designed primarily for touch screen mobile devices such as smart phone and tablets.

Since Android devices are usually battery-powered, Android is designed to manage processes to keep power consumption at a minimum. When an application is not in use the system suspends its operation so that, while available for intermediate use rather than closed, it does not use battery power or CPU resources. Android manages the application stored in memory automatically: when memory is low, the system will begin invisibly and automatically closing inactive processes, starting with those that have been inactive for the longest amount of time.

Android devices incorporate many optical hardware components, including video cameras, GPS, orientation sensors, dedicated gaming controls, accelerometers, gyroscopes, barometers, magnetometers, proximity sensors, thermometers and touch screens. Some hardware components are not required, but became standard in certain classes of devices, such as smart phone, and additional requirements apply if they are present. Some other hardware was initially required, but those requirements have been relaxed or eliminated altogether. For example, as Android was developed initially as a phone OS, hardware such as microphones were required, while over time the phone function became optional. Android used to require an autofocus camera, which was relaxed to a fixed-focus camera if present at all, since the camera was dropped as a requirement entirely when Android started to be used on set-top boxes.

Android's source code is released by Google under an open source license, and its open nature has encouraged a large community of developers and enthusiasts to use the open-source code as a foundation for community-driven projects, which deliver updates to older devices, add new features for advanced users or bring Android to devices originally shipped with other operating systems. These

community-developed releases often bring new features and updates to devices faster than through the official manufacturer/carrier channels, with a comparable level of quality, provide continued support for older devices that no longer receive official updates; or bring Android to devices that were officially released running other operating systems, such as the HP Touchpad.

### 3.2 SDLC Model

The Software Development Life Cycle Model (SDLC) we have used in this project is Agile Model for developing our application. Agile SDLC model is a combination of iterative and incremental process models with focus on process adaptability and customer satisfaction by rapid delivery of working software product. Agile Methods break the product into small incremental builds. These builds are provided in iterations. Each iteration typically lasts from about one to three weeks. The goal of agile software development is to make the development process flexible and slim line.

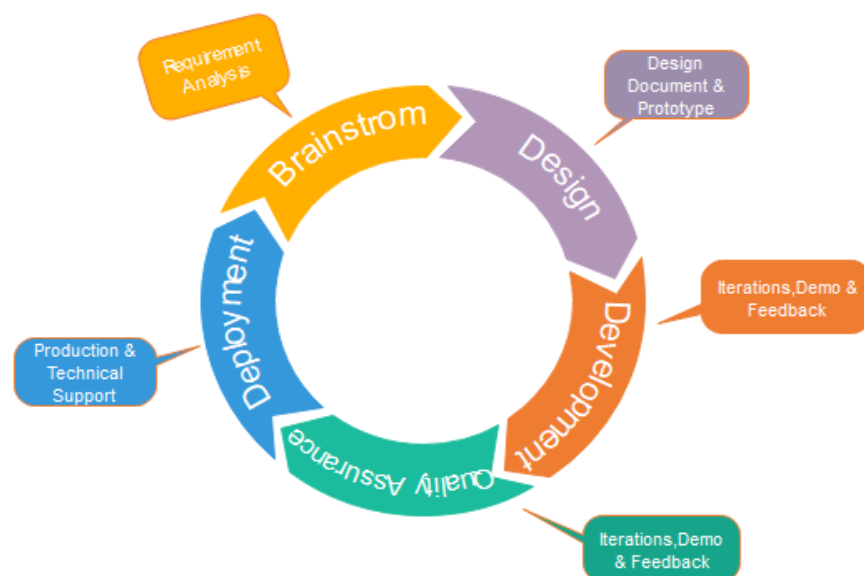


Fig 3.1: Agile Model

Following are the phases in the agile model are as follows:

- Requirements gathering
- Design the requirements
- Construction/ iteration
- Testing/ Quality assurance
- Deployment
- Feedback

### **1. Requirements gathering:**

In this phase, we must define the requirements. We should explain business opportunities and plan the time and effort needed to build the project. Based on this information, you can evaluate technical and economic feasibility.

### **2. Design the requirements:**

When we have identified the project, work with stakeholders to define requirements. We can use the user flow diagram or the high-level UML diagram to show the work of new features and show how it will apply to our existing system.

### **3. Construction/ Iteration:**

When the team defines the requirements, the work begins. Designers and developers start working on their project, which aims to deploy a working product. The product will undergo various stages of improvement, so it includes simple, minimal functionality.

### **4. Testing:**

In this phase, the Quality Assurance team examines the product's performance and looks for the bug. We prefer to do unit testing for examining the performance of the application.

### **5. Deployment:**

In this phase, the issues a product for the user's work environment.

### **6. Feedback:**

This is good include customer feedback about the product, improvement in product quality better decision making timelines of information, expediting activities, improve accuracy of operation, better documentation and record keeping, faster retrieval of information, better employ moral.

### 3.3 Software Requirement Specification

A software requirements specification (SRS) is a description of a software system to be developed. It is modeled after business requirements specification (CONOPS), also known as a stakeholder requirements specification (SRS). The software requirements specification lays out functional and non-functional requirements, and it may include a set of use cases that describe user interactions that the software must provide to the user for perfect interaction.

Software requirements specification establishes the basis for an agreement between customers and contractors or suppliers on how the software product should function (in a market-driven project, these roles may be played by the marketing and development divisions). Software requirements specification is a rigorous assessment of requirements before the more specific system design stages, and its goal is to reduce later redesign. It should also provide a realistic basis for estimating product costs, risks, and schedules. Used appropriately, software requirements specifications can help prevent software project failure.

The software requirements specification document lists sufficient and necessary requirements for the project development. To derive the requirements, the developer needs to have clear and thorough understanding of the products under development. This is achieved through detailed and continuous communications with the project team and customer throughout the software development process.

The SRS may be one of contract deliverable data item descriptions or have other forms of organizationally-mandated content.

#### **Characteristics of SRS:**

**Correct** - An SRS is correct if, and only if, every requirement stated therein is one that the software shall meet. Traceability makes this procedure easier and less prone to error.

**Unambiguous** - An SRS is unambiguous if, and only if, every requirement stated therein has only one interpretation. As a minimum, this requires that each characteristic of the final product be described using a single unique term.

**Verifiable** – It is verifiable if there exists some finite cost-effective process with which a person or machine check whether software product meets requirements.

**Consistent** - Consistency refers to internal consistency. If an SRS does not agree with some higher-level document, such as a system requirements specification, then it is not correct. An SRS is internally consistent if, and only if, no subset of individual requirements described in it conflict.

**Modifiable** – SRS is said to be modifiable if its structure and style are such that any changes to the requirements can be made easily, completely and consistently while retaining the structure and style.

**Traceable** – SRS is said to be traceable if the origin of each of its requirements is clear and it facilitates the referencing of each requirement in future enhancement.

**Ranked for importance or stability** – SRS is ranked for importance or stability if each requirement in it has an identifier to indicate either the importance or stability of that particular requirement.

### 3.3.1 User Requirements

The software requirements specification is produced at the culmination of the analysis task. The function and performance allocated to the software as a part of system engineering and refined by establishing a complex information description, detailed functional and behavioral description, and indication of performance requirements and design constraints, appropriate validation criteria and other data pertinent to requirements.

The major requirement included in this application is to detect the duplicate the data. This is the highest level abstraction of the requirement, this to be converted into lower level requirement language in detail. The programmer has to correctly have understood.

### 3.3.2 Software Requirements

- Operating System :Windows 10/ Ubuntu
- Database :Room Database, Firebase Database
- Programming language :Java
- Tools :Android Studio (IDE), JDK 1.8 Software, Star UML

### 3.3.3 Hardware Requirements

- CPU type :Intel Pentium 5
- Ram size :8 GB
- Hard disk capacity :200 GB
- Android Device (above 4.1 version- Jelly Bean )

## 3.4 Android Studio Installation Procedure

Setting up Android development environment takes some time at first. It helps to make sure you don't do anything wrong to save yourself from the agony of doing the whole process again.

You're required to have [Windows XP](#) or later, or Mac OS X 10.5.8 or a later version to start Android application development process.

Then, there are four tools that you will need and they are available on the Internet for free:

- Java JDK5 or JDK6
- Android SDK

### Step 1: Setup Java Development Kit (JDK)

You can [download the JDK](#) and install it, which is pretty easy. After that, you just have to set PATH and JAVA\_HOME variables to the folder where you have **java** and **javac**.

**Note for Windows Users:** If you installed the JDK in C:\jdk1.6.0\_15 then you will have to add the following two lines in your command prompt

C:\autoexec.bat file.

Set PATH=C:\jdk1.6.0\_15\bin;%PATH%

Set JAVA\_HOME=C:\jdk1.6.0\_15

### Step 2: Downloading and setting up Android Studio

Google provides Android Studio for the Windows, Mac OS X, and Linux platforms. You can download this software from the Android Studio homepage. (You'll also find the traditional SDKs, with Android Studio's command-line tools, available from the Downloads page.)



## Installing Android studio:

Launch `android-studio-ide-191.6010548-windows.exe` to start the installation process. The installer responded by presenting the Android Studio Setup dialog box shown below.



Fig.3.2: Set up Android Studio

Clicking Next took me to the following dialog box, which gives you the option to decline installing the Android SDK (included with the installer) and an Android Virtual Device.

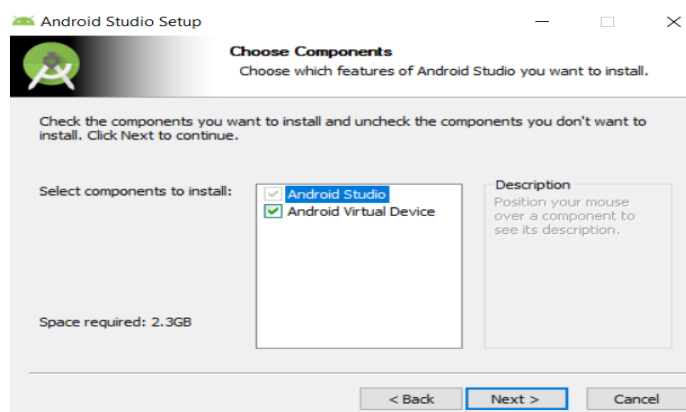


Fig.3.3: Do you want to install the Android SDK and AVD

After clicking next, you'll be taken to configuration settings.

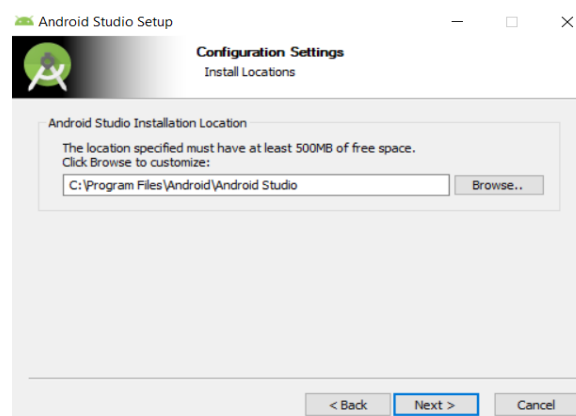


Fig.3.4: Set the Android Studio installation locations

Change the location or accept the default locations and click Next. The installer defaults to creating a shortcut for launching this program, or you can choose to decline. I recommend that you create the shortcut, and then click the Install button to begin installation.

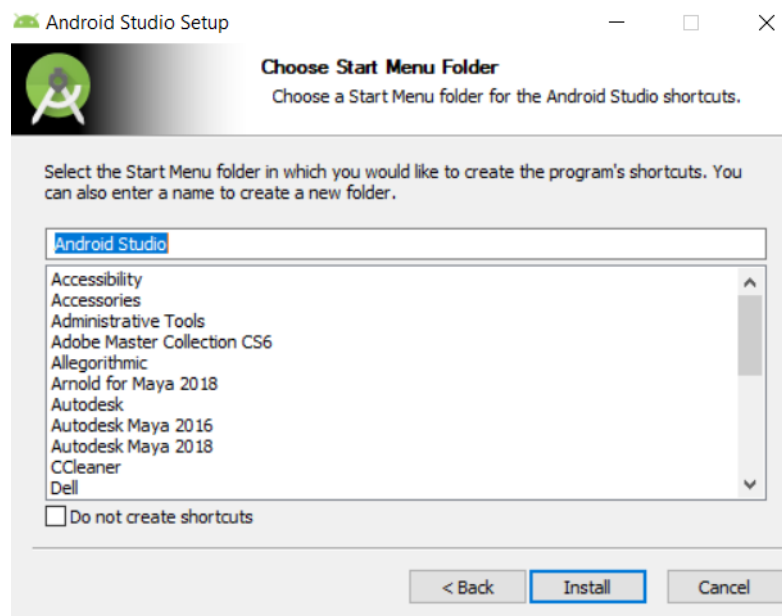


Fig.3.5: Create a new shortcut for Android Studio

The resulting dialog box shows the progress of installing Android Studio and the Android SDK. Clicking the Show Details button will let you view detailed information about the installation progress.

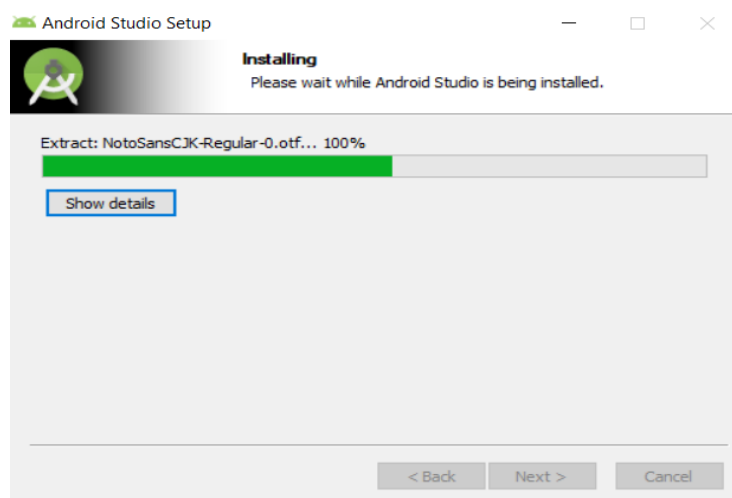


Fig.3.6: Installation in progress

The dialog box will inform you when installation has finished. When you click Next, you should see the following:

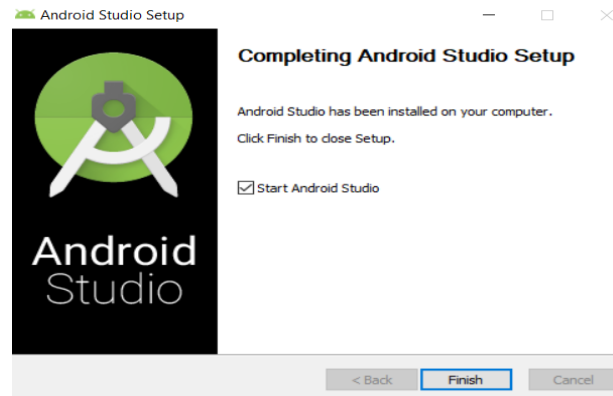


Fig.3.7: Leave the Start Android Studio check box checked to run this software.

To complete your installation, leave the Start Android Studio box checked and click Finish.

### Running Android Studio

Android Studio presents a splash screen when it starts running:



Fig 3.8: Android Studio's start screen

On your first run, you'll be asked to respond to several configuration-oriented dialog boxes. The first dialog box focuses on importing settings from any previously installed version of Android Studio.

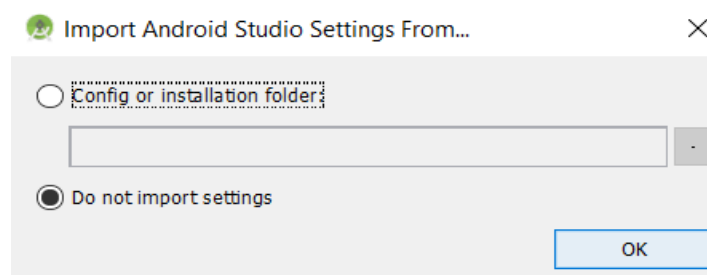


Fig.3.9: Import settings

If you're like me, and don't have a previously installed version, you can just keep the default setting and click OK. Android Studio will respond with a slightly enhanced version of the splash screen, followed by the Android Studio Setup Wizard dialog box:

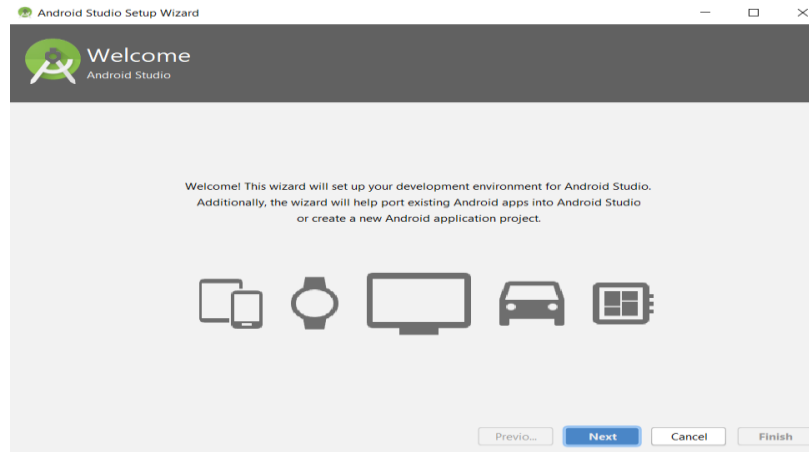


Fig.3.10: Validate your Android SDK and development environment setup

When you click Next, the setup wizard invites you to select an installation type for your SDK components. For now I recommend you keep the default standard setting.

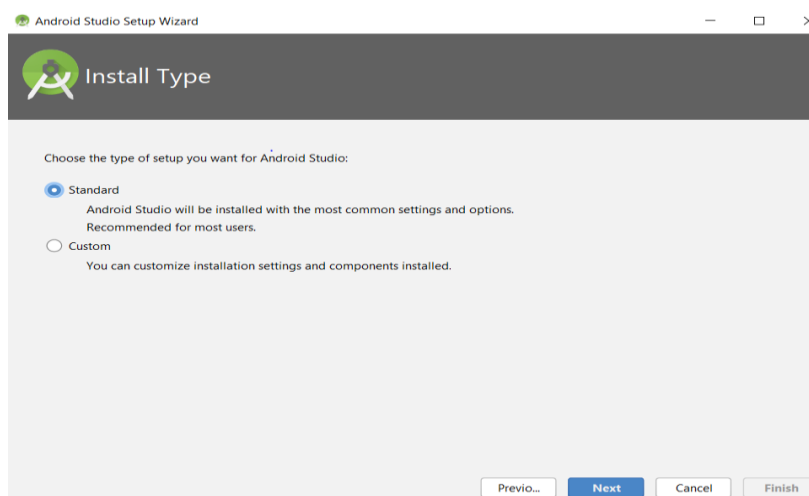


Fig.3.11: Choose an installation type

Click Next and verify your settings, then click Finish to continue.

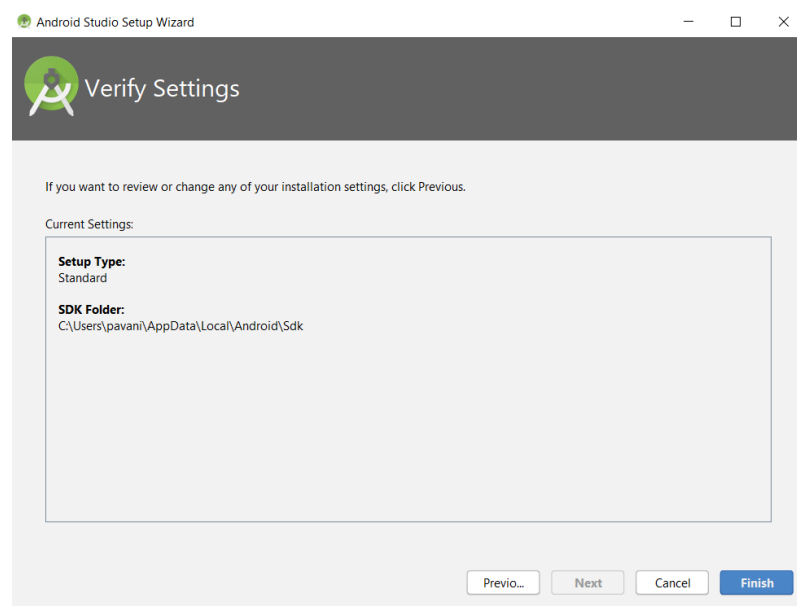


Fig.3.12: Review settings

The wizard will download and unzip various components. Click Show Details if you want to see more information about the archives being downloaded and their contents.

Finally, click Finish to complete the wizard. You should see the Welcome to Android Studio dialog box:

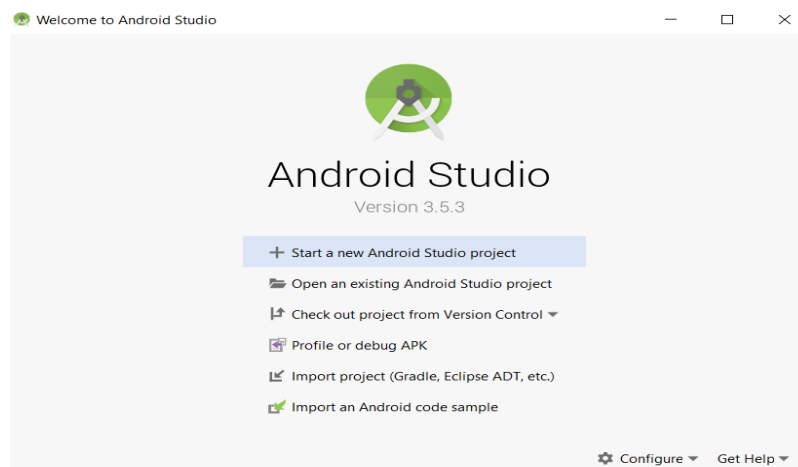


Fig.3.13: Welcome to Android Studio

You'll use this dialog to start up a new Android Studio project, work with an existing project, and more. You can access it anytime by double-clicking the Android Studio shortcut on your desktop.

## Starting a new project

From our setup so far, you should still have Android Studio running with the Welcome to Android Studio dialog box. From here, click Start a new Android Studio project. Click Next, and you will be given the opportunity to choose a template for your app's main activity. For now we'll stick with Empty Activity. Select this template and click Next.

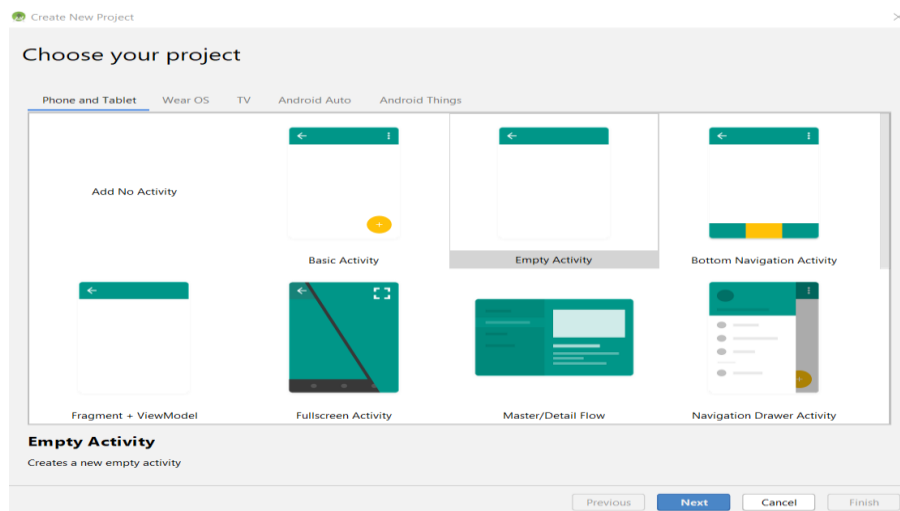


Fig.3.14: Specify an activity template

Next you'll customize the activity:

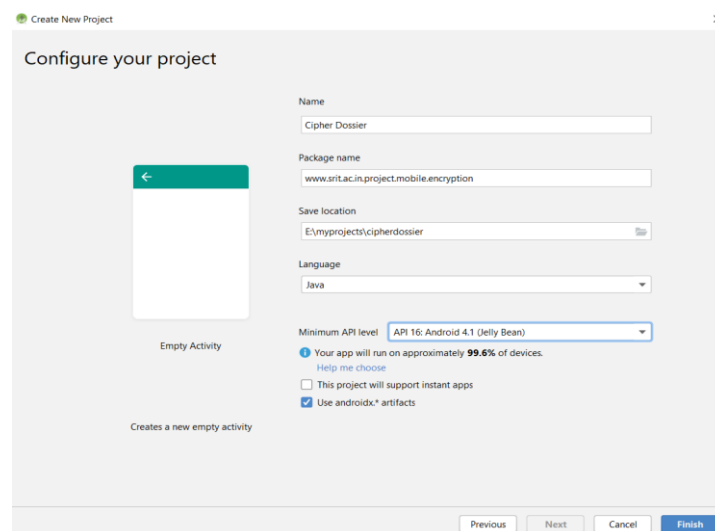


Fig.3.15: Customize your activity

Enter *Cipher Dossier* as the application name and *www.srit.ac.in.mobile.encryption* as the company domain name. You should then see *E:\myprojects\cipherdossier* as the project location, select your target devices.

Android Studio lets you select *form factors*, or categories of target devices, for every app you create. I would have preferred to keep API 16: Android 4.1(JellyBean) minimum SDK setting (under Phone and Tablet). Then click on finish. Android Studio will respond that it is creating the project, and then take you to the project workspace.

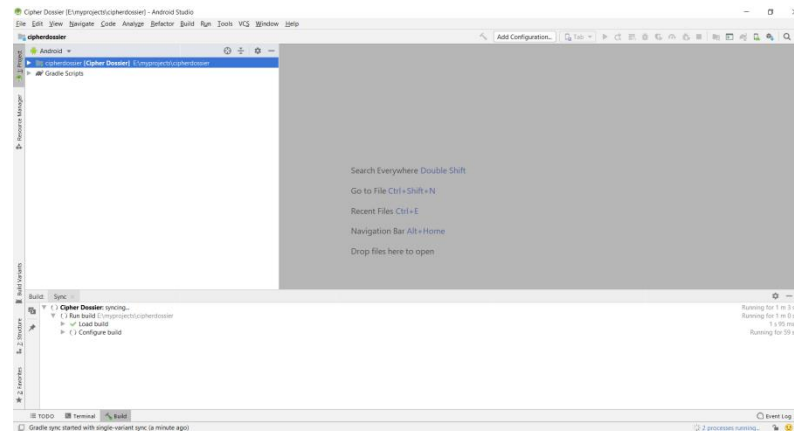


Fig.3.16: Android Studio workspace

The project workspace is organized around a menu bar, a tool bar, a work area, additional components that lead to more windows, and a status bar. Also note the Tip of the Day dialog box, which you can disable if you like.

### The project and editor windows

When you enter the project workspace, W2A is identified as the current project, but you won't immediately see the project details. After a few moments, these details will be appearing in two new windows.

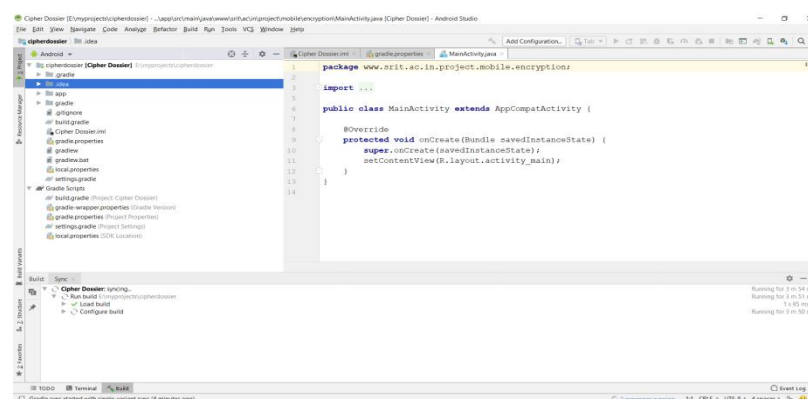


Fig.3.17: The project and editor windows

### 3.5 Summary

The Cipher Dossier application whose specifications have been provided Simple but it is very helpful for the users to hide their data like images, videos, messages, etc. We also see how requirements gathering and design are very critical in software development to make sure that the project solves the right problem in right approach.



## **CHAPTER-4**

### **DESIGN**

#### **4.1 Introduction**

System design is the solution to the creation of a new system. This phase is composed of several systems. This phase focuses on the detailed implementation of the feasible system. It emphasis on translating design specifications to performance specification is system design. System design has two phases of development logical and physical design.

During logical design phase the analyst describes inputs (sources), outputs (destinations), databases (data stores) and procedures (data flows) all in a format that meats the uses requirements. The analyst also specifies the user needs and at a level that virtually determines the information flow into and out of the system and the data resources. Here the logical design is done through data flow diagrams and database design.

The physical design is followed by physical design or coding. Physical design produces the working system by defining the design specifications, which tell the programmers exactly what the candidate system must do.

The programmers write the necessary programs that accept input from the user, perform necessary processing on accepted data through call and produce the required report on a hard copy or display it on the screen.

#### **4.2 UML DIAGRAMS**

UML stands for Unified Modeling Language. UML is a standardized general purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta- model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, visualization, constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems. The Unified Modeling Language represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The Unified Modeling Language is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

**Goals:**

The primary goals in the design of the UML are as follows:

- Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
- Provide extendibility and specialization mechanisms to extend the core concepts.
- Be independent of particular programming languages and development process.
- Provide a formal basis for understanding the modeling language.
- Encourage the growth of the Object Oriented tools market.
- Support higher level development concepts such as collaborations, frameworks, patterns and components.

**4.2.1 Use case Diagram**

A Use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases.

The user has to register by entering all details and login into the android application and store all the data by giving a key using which all data will be encrypted using AES algorithm. By using the same password only data can be retrieved.

The following is the Use Case diagram:

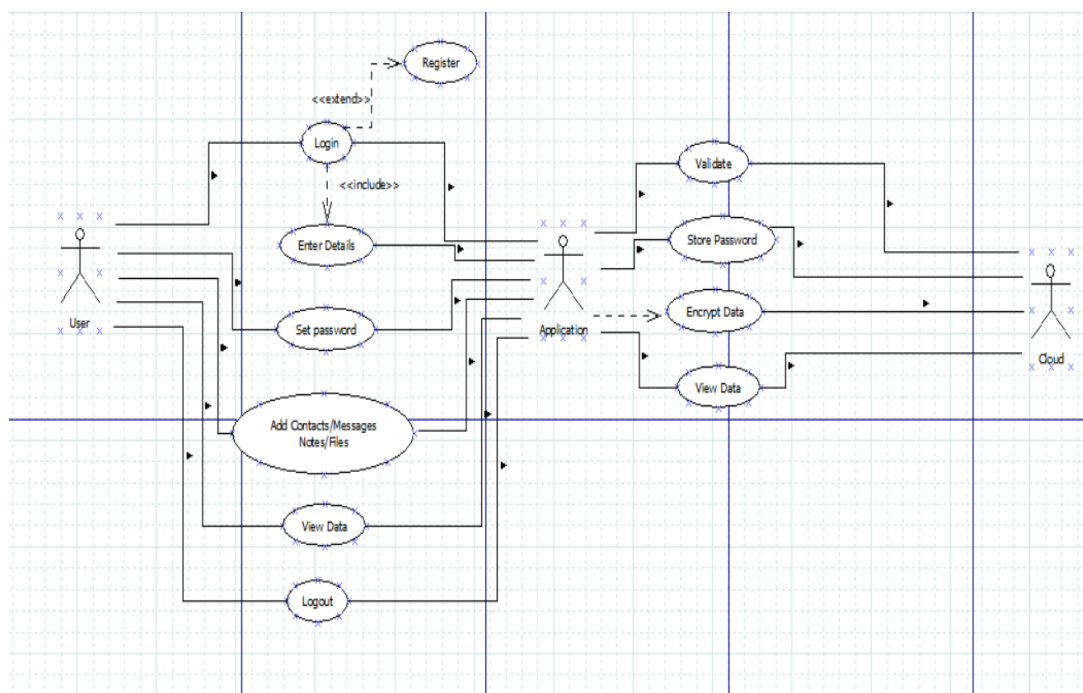


Fig 4.1: Use case Diagram

#### 4.2.2 Sequence Diagram

A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called event diagrams or event scenarios.

A sequence diagram shows, as parallel vertical lines (*lifelines*), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.

In the sequence diagram, first the user registers by entering all details and login into the android application sequentially and store all the data by giving a key using which all data will be encrypted using AES algorithm. By the same login credentials using the same key only data can be retrieved.

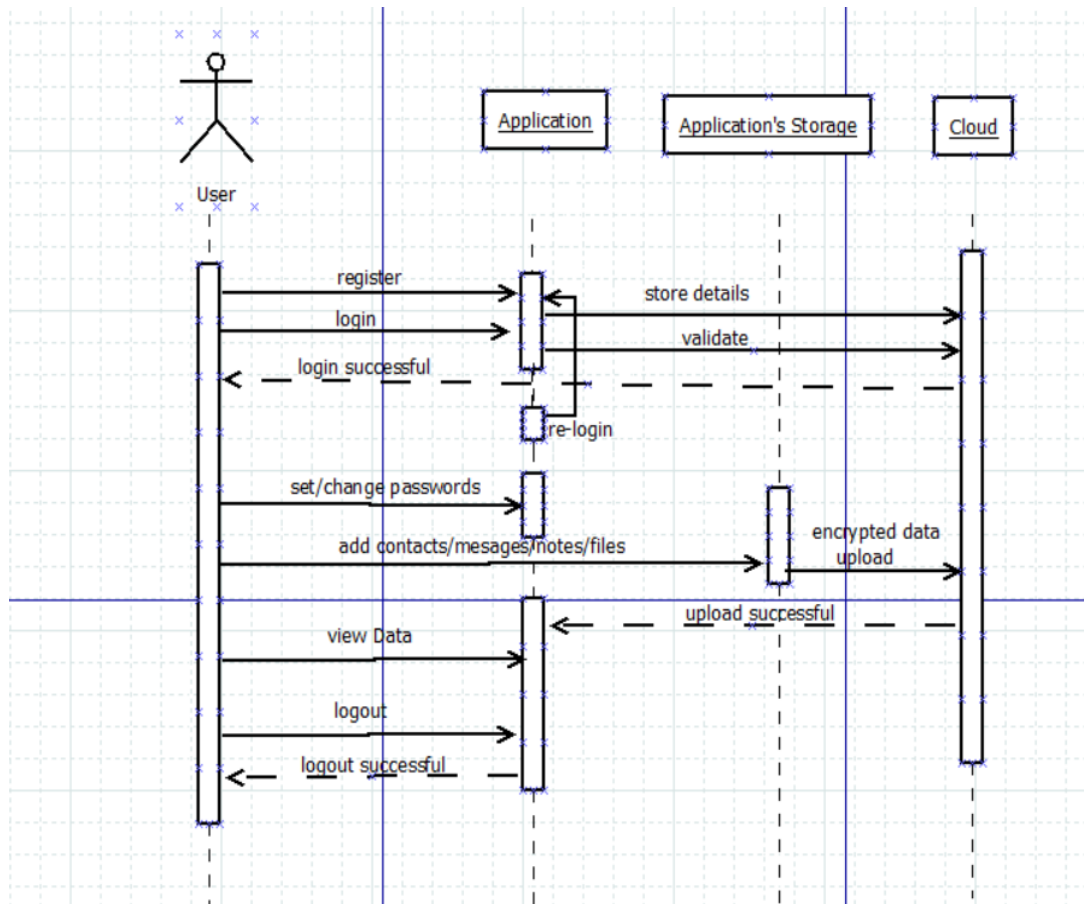


Fig 4.2: Sequence Diagram

### 4.2.3 Class Diagram

In software engineering, a **class diagram** in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

In the diagram, classes are represented with boxes which contain three parts:

- The top part contains the name of the class
- The middle part contains the attributes of the class
- The bottom part gives the methods or operations the class can take or undertake

There are different classes like user, application, messages, contacts, files, cloud each with different properties. All the data uploaded will be encrypted using AES algorithm and stored in cloud.

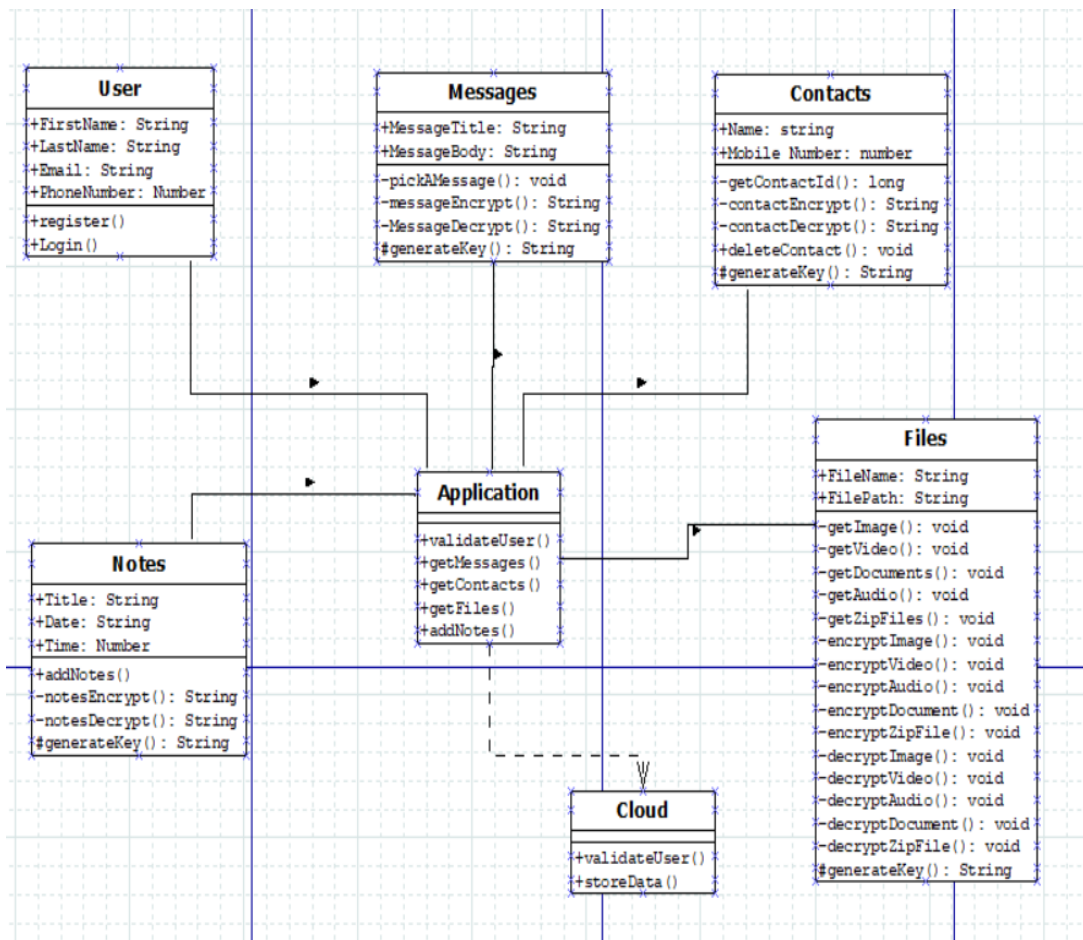


Fig: 4.3: Class Diagram

#### 4.2.4 State Chart Diagram

State chart diagram is one of the five UML diagrams used to model the dynamic nature of a system. They define different states of an object during its lifetime and these states are changed by events. State chart diagrams are useful to model the reactive systems. Reactive systems can be defined as a system that responds to external or internal events.

State chart diagram describes the flow of control from one state to another state. The most important purpose of State chart diagram is to model lifetime of an object from creation to termination like register, login, logout and different states where user performs the operations at different states.

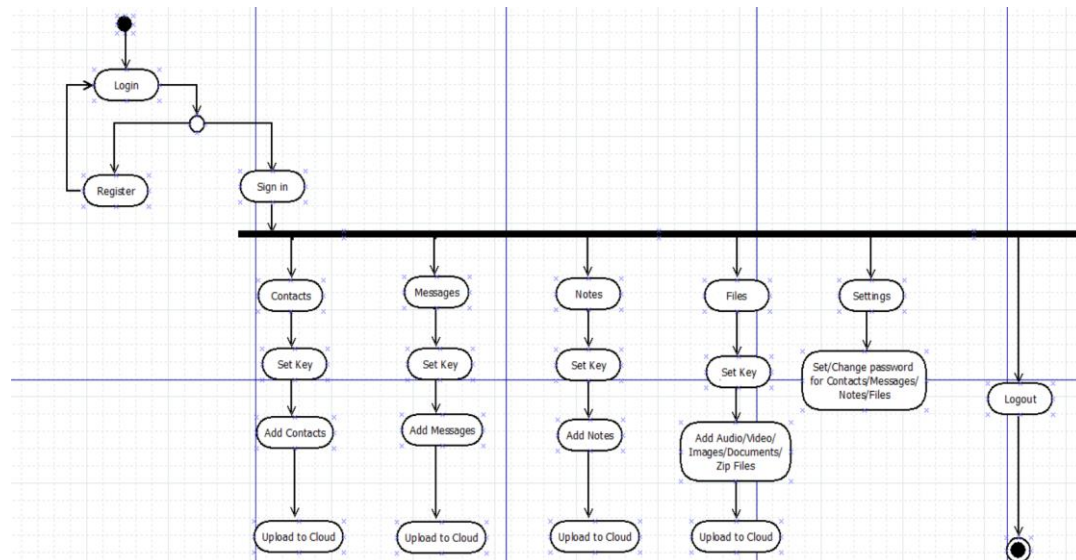


Fig 4.4: State Chart Diagram

## 4.2.5 Activity Diagram

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams are intended to model both computational and organizational processes (i.e., workflows), as well as the data flows intersecting with the related activities.

Activity diagram describes the flow of control from one activity to another activity. The most important purpose of activity diagram is to model lifetime of an object from creation to termination like register, login, logout and different states where user performs the operations at different activities.

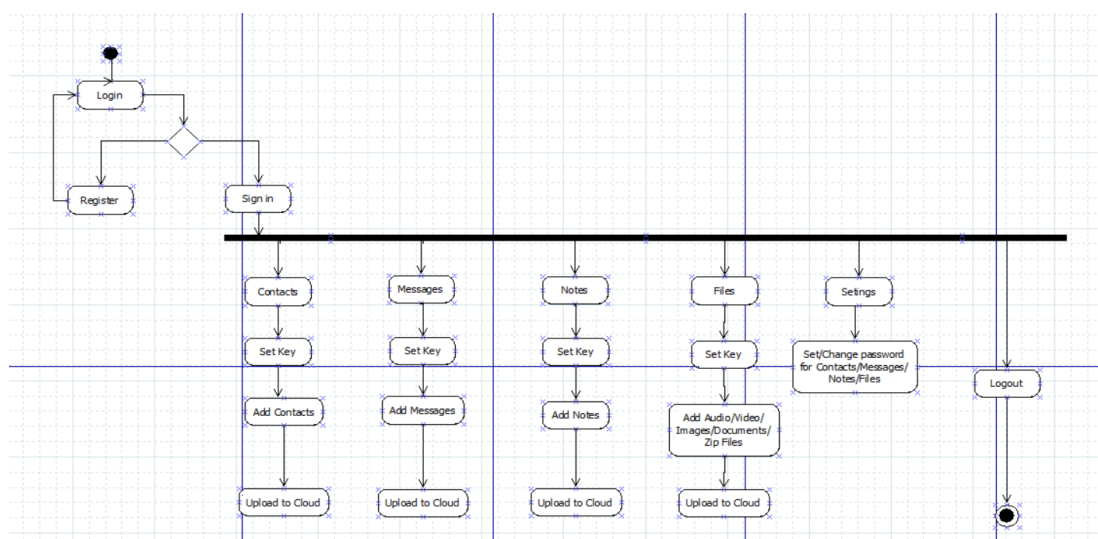


Fig 4.5: Activity Diagram



• • • • •



### 4.2.7 Component Diagram

In Unified Modeling Language (UML), a component diagram depicts how components are wired together to form larger components or software systems. They are used to illustrate the structure of arbitrarily complex systems.

In this diagram all the components like contacts, messages, files, notes are together wired to form a large component which is application.

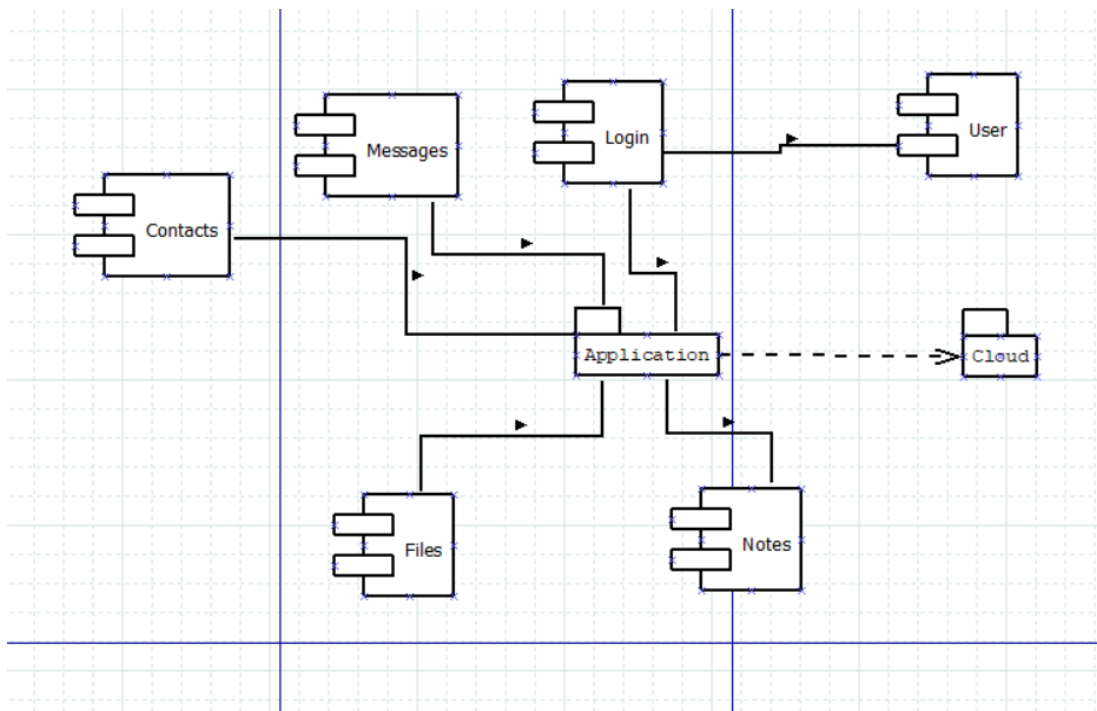


Fig 4.7: Component Diagram

### 4.2.8 Collaboration Diagram

A collaboration diagram, also called a communication diagram or interaction diagram, is an illustration of the relationships and interactions among software objects in the Unified Modeling Language (UML). Collaboration diagrams show objects, their links, and their messages. They can also contain simple class instances and class utility instances. Each collaboration diagram provides a view of the interactions or structural relationships that occurs between objects and object-like entities in the current model.

This diagram shows the links between the user, application, cloud and the interaction between them.



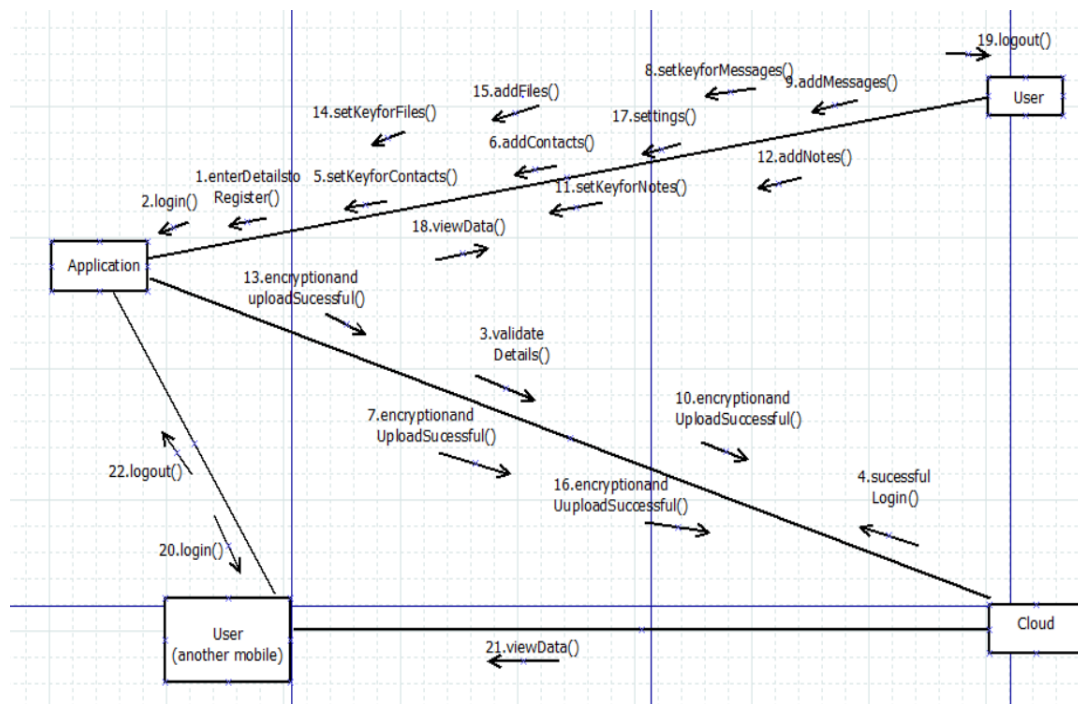


Fig 4.8: Collaboration Diagram

### 4.3 Module design and organization

#### User Module:

The User has to sign in the account using their email authentication, otherwise the application cannot be accessed. While signing the account, user has to enter his name, email and password, mobile number, if account does not exist then needed to create an account. User can add their data in different slots provided like contacts, messages, notes, files. User can upload their data in to the cloud. If the user lost his mobile, he can get his data by logging into the mobile. If the user successfully uploads his data to cloud he can access the data from any mobile.

#### Admin Module:

The admin has access to cloud. He can help the users.

### 4.4 Summary

Design of this application provides clear explanation of all modules and their functionalities with their UML diagrams.

## **CHAPTER-5**

### **IMPLEMENTATION & RESULTS**

#### **5.1 Introduction**

After designing the new system, the whole system is required to be converted into computer understanding language. Coding the new system into computer programming language does this.

It is an important stage where the defined procedures are transformed into control specifications by the help of a computer language. This is also called the programming phase in which the programmer converts the program specifications into computer instructions, which we refer as programs. The programs coordinate the data movements and control the entire process in a system.

It is generally felt that the programs must be modular in nature. This helps in fast development, maintenance and future change, if required.

The validity and proper functionality of all the modules of the developed application is assured during the process of implementation. Implementation is the process of assuring that the information system is operational and then allowing user to take over its operation for use and evaluation.

Implementation is the stage in the project where the theoretical design is turned into a working system. The implementation phase constructs, installs and operated the new system. The most crucial stage in achieving a new successful system is that it works effectively and efficiently.

#### **5.2 Explanation of Key functions**

##### **5.2.1. Firebase**

Firebase is a [mobile](#) and [web application](#) development platform developed by Firebase, Inc. in 2011, then acquired by [Google](#) in 2014. As of October 2018, the Firebase platform has 18 products, which are used by 1.5 million apps [5].

Firebase is a mobile platform that helps you quickly develop high-quality apps, grow your user base, and earn more money. Firebase is made up of complementary features that you can mix-and-match to fit your needs, with Google Analytics for Firebase at the core. You can explore and integrate Firebase services in your app directly from Android Studio using the Assistant window.

#### **5.2.1.1 Services of Firebase**

##### **1. Firebase Analytics**

Firebase Analytics is a cost-free app measurement solution that provides insight into app usage and user engagement.

##### **2. Firebase Cloud Messaging**

Formerly known as Google Cloud Messaging (GCM), Firebase Cloud Messaging (FCM) is a cross-platform solution for messages and notifications for Android, iOS, and web applications, which as of 2016 can be used at no cost.

##### **3. Firebase Auth**

Firebase Auth is a service that can authenticate users using only client-side code. It supports social login providers Facebook, GitHub, Twitter and Google (and Google Play Games). Additionally, it includes a user management system whereby developers can enable user authentication with email and password login stored with Firebase.

##### **4. Real-time database**

Firebase provides a real-time database and backend as a service. The service provides application developers an API that allows application data to be synchronized across clients and stored on Firebase's cloud.

The database is also accessible through a REST API and bindings for several JavaScript frameworks such as AngularJS, React, Ember.js and Backbone.js. The REST API uses the Server-Sent Events protocol, which is an API for creating HTTP connections for receiving push notifications from a server. Developers using the real-time database can secure their data by using the company's server-side-enforced security rules. Cloud Firestore which is Firebase's next generation of the Real-time Database was released for beta use.

## 5. Firebase Storage

Firebase Storage provides secure file uploads and downloads for Firebase apps, regardless of network quality. The developer can use it to store images, audio, video, or other user-generated content. Firebase Storage is backed by Google Cloud Storage.

## 6. Firebase Hosting

Firebase Hosting is a static and dynamic web hosting service that launched on May 13, 2014. It supports hosting static files such as CSS, HTML, JavaScript and other files, as well as support through Cloud Functions. The service delivers files over a content delivery network (CDN) through HTTP Secure (HTTPS) and Secure Sockets Layer encryption (SSL). Firebase partners with fast, a CDN, to provide the CDN backing Firebase Hosting. The company states that Firebase Hosting grew out of customer requests; developers were using Firebase for its real-time database but needed a place to host their content.

### 5.2.2. Room Database

The Room persistence library provides an abstraction layer over SQLite to allow for more robust database access while harnessing the full power of SQLite. The library helps you create a cache of your app's data on a device that's running your app. This cache, which serves as your app's single source of truth, allows users to view a consistent copy of key information within your app, regardless of whether users have an internet connection. Steps for using Room Database:

#### Step-1: Add Dependencies

```
// Room components
implementation "androidx.room:room-runtime:2.2.3"
annotationProcessor "androidx.room:room-

// Lifecycle components
implementation "androidx.lifecycle:lifecycle-
extensions:2.2.0"
annotationProcessor "androidx.lifecycle:lifecycle-
common-java8:2.1.0"
```

**Step-2: Create Entity**

There are actually two requirements for the entity:

- **@Entity** annotation on the class
- At least one (\*) **@PrimaryKey** field (if it's non primitive, should be annotated with **@NonNull**)

**Step-3: Create Data Access Object (DAO)**

DAO tells our database how to put the data. It includes the following methods:

- **@Insert**, **@Update**, **@Delete** for proper actions: inserting, updating and deleting records
- **@Query** for creating queries—we can make select from the database (e.g. get all repos)

**Step-4: Create Database**

**@Database** annotation with **entities** to our model classes and **version** of our database. Then we have `getInstance()` method which is nothing more than a simple singleton pattern.

**Step-5: Instantiate the Database**

We do it so by writing the following code :

```
RepoDatabase.getInstance(context).getRepoDao().insert(new Repo(1, "Cool Repo Name", "url"));
```

### 5.2.3 RecyclerView

The RecyclerView widget is a more advanced and flexible version of ListView. In the RecyclerView model, several different components work together to display your data. The overall container for your user interface is a RecyclerView object that you add to your layout. The RecyclerView fills itself with views provided by a *layout manager* that you provide. You can use one of our standard layout managers (such as `LinearLayoutManager` or `GridLayoutManager`), or implement your own.

The views in the list are represented by *view holder* objects. These objects are instances of a class you define by extending `RecyclerView.ViewHolder`. Each view

holder is in charge of displaying a single item with a view. For example, if your list shows music collection, each view holder might represent a single album. The RecyclerView creates only as many view holders as are needed to display the on-screen portion of the dynamic content, plus a few extra. As the user scrolls through the list, the RecyclerView takes the off-screen views and rebinds them to the data which is scrolling onto the screen.

Android Studio version 3.5.3 supports RecyclerView without any dependencies

The following layout uses RecyclerView as the only view for the whole layout:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.recyclerview.widget.RecyclerView
    tools:listitem="@layout/contacts_item"
    app:layoutManager="androidx.recyclerview.widget.LinearLayoutManager"
    android:layout_alignParentTop="true"
    android:id="@+id/contacts_RecyclerView"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

</androidx.recyclerview.widget.RecyclerView>
```

Fig 5.1: Code in XML file for recycler view

#### 5.2.4 AES Algorithm

AES stands for Advanced Encryption Standard Algorithm which is a block cipher technique. We use the AES instances to encrypt/decrypt the data. Consider the following code snippet in which a method for encrypting the contacts is present.

```
private String contactEncrypt(String message, String passKey) throws Exception {
    SecretKeySpec key = generateKey(passKey);
    Cipher cipher = Cipher.getInstance("AES");
    cipher.init(Cipher.ENCRYPT_MODE, key);
    byte[] encryptedMsg = cipher.doFinal(message.getBytes());
    return Base64.encodeToString(encryptedMsg, Base64.DEFAULT);
}
```

Fig 5.2: Encrypt method

In the same way the decryption method is present where the data is decrypted which is as follows.

```
private String contactDecrypt(String encStr, String passKey) throws Exception {
    SecretKeySpec key = generateKey(passKey);
    Cipher cipher = Cipher.getInstance("AES");
    cipher.init(Cipher.DECRYPT_MODE, key);
    byte[] decMsg = Base64.decode(encStr, Base64.DEFAULT);
    byte[] decodedMsg = cipher.doFinal(decMsg);
    return new String(decodedMsg);
}
```

Fig 5.3: Decrypt method

Using these methods the contacts are encrypted and decrypted. In the same way using the methods like these other data is also encrypted and decrypted.

## 5.3 Method of Implementation

### 5.3.1 Output Screens

**Screen-1:** It is the login page of the application.

Fig 5.4: Login Page

**Screen-2:** It is the registration page of the application where the user needs to fill it with his/her details and set a password.

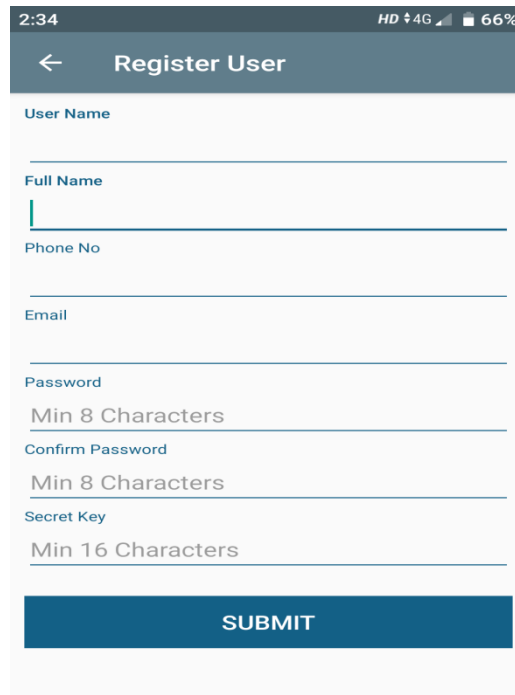
A screenshot of a mobile application's registration page. The status bar at the top shows the time 2:34, HD 4G signal, and 66% battery. The page has a dark blue header with a back arrow and the title "Register User". Below the header, there are several input fields: "User Name", "Full Name", "Phone No", "Email", "Password" (with a "Min 8 Characters" hint), "Confirm Password" (with a "Min 8 Characters" hint), and "Secret Key" (with a "Min 16 Characters" hint). At the bottom, there is a dark blue button labeled "SUBMIT".

Fig 5.5: Registration Page

**Screen-3:** It is the home screen of the application where we can see different slots like contacts, messages, notes, files. And also it contains settings and logout option.

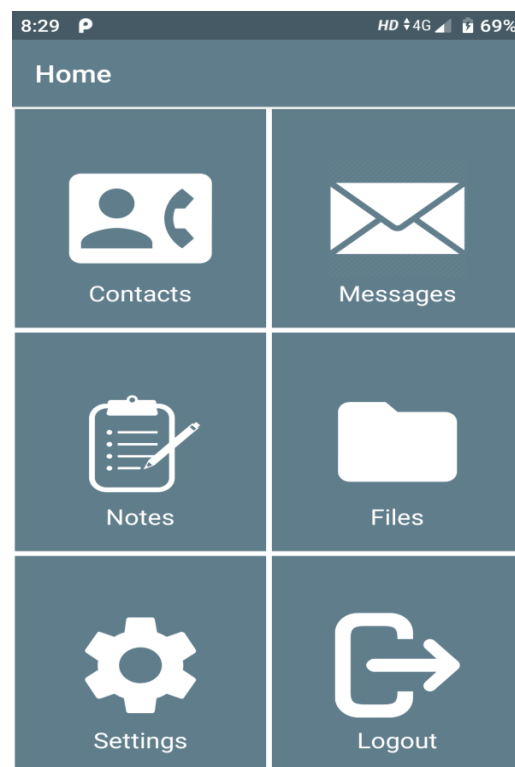


Fig 5.6: Home Activity



**Screen-4:** It is settings activity to set keys for contacts, messages, notes and files.

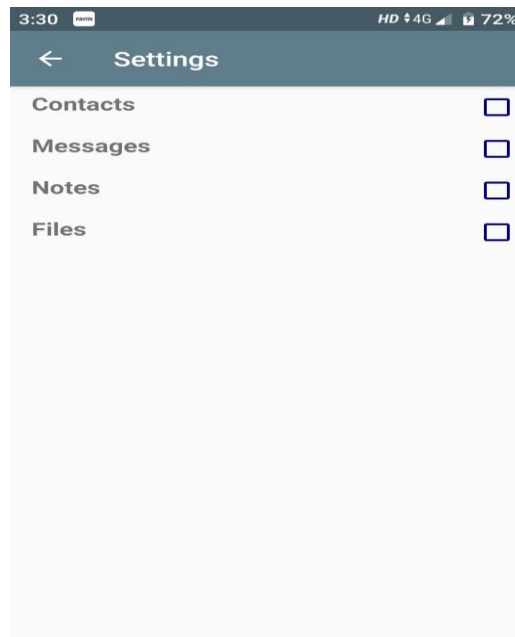


Fig 5.7: Settings Activity

**Screen-5:** So at first we need to enter the password for enabling that options. After that we need to enter the password if we enable that option.

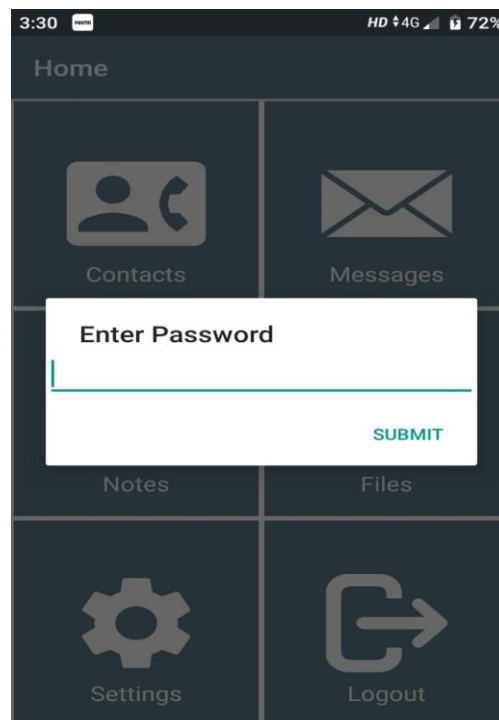


Fig 5.8: Entering Key

**Screen-6:** It is contacts activity. It is empty and need to add the contacts here.

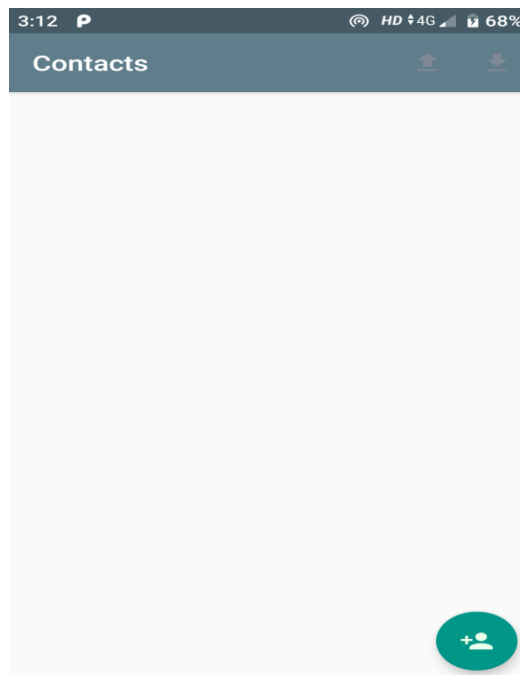


Fig 5.9: Contacts Activity

**Screen -7:** In this screenshot we can see a contact have been added and we have even a delete icon there. We even have two options like upload and download icons available.

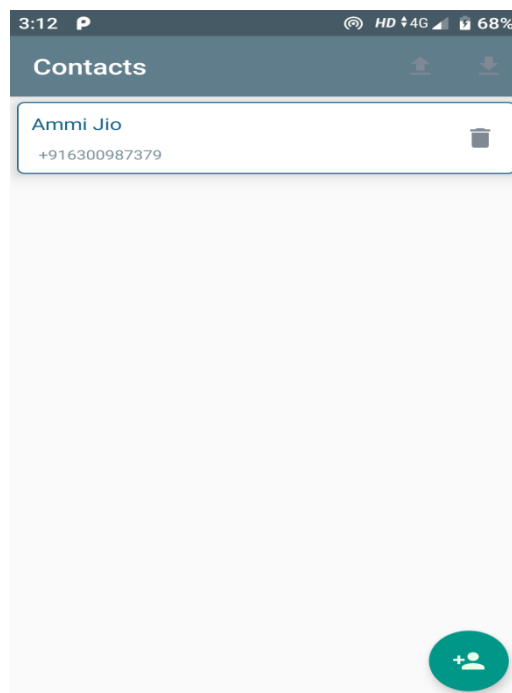


Fig 5.10: After adding Contacts

**Screen-8:** The below screenshots resembles the messages activity after adding a message.

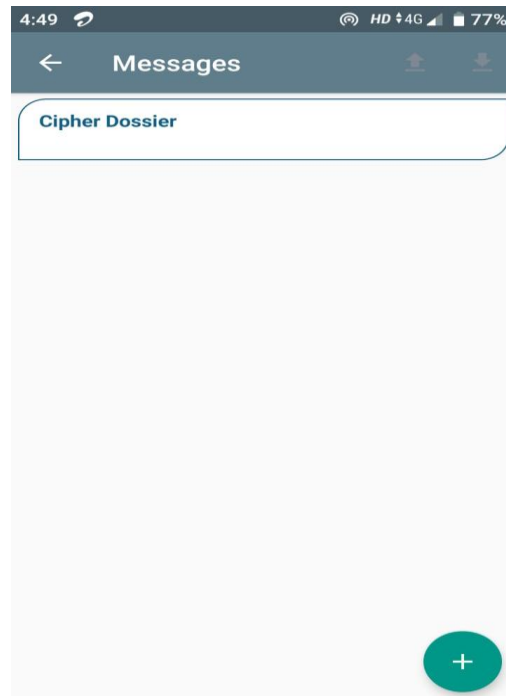


Fig 5.11 Messages activity

**Screen-9:** The below screenshot is notes activity when a notes is added. To add a notes we need to click on that “+” floating action button.

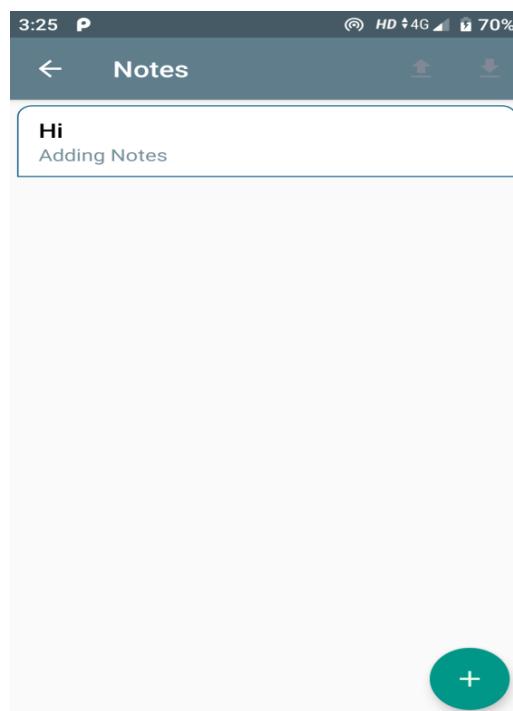


Fig 5.12: Notes activity

**Screen-10:** Here we can add a notes which contains date, time, title and description. After adding we have to click on floating action button.

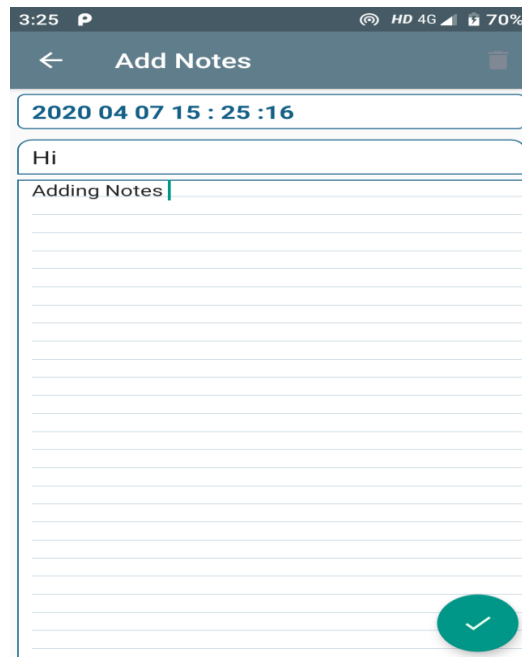


Fig 5.13: Adding Notes

**Screen-11:** The below screenshot is files activity where we can add images, documents, audios, videos, and zip files.

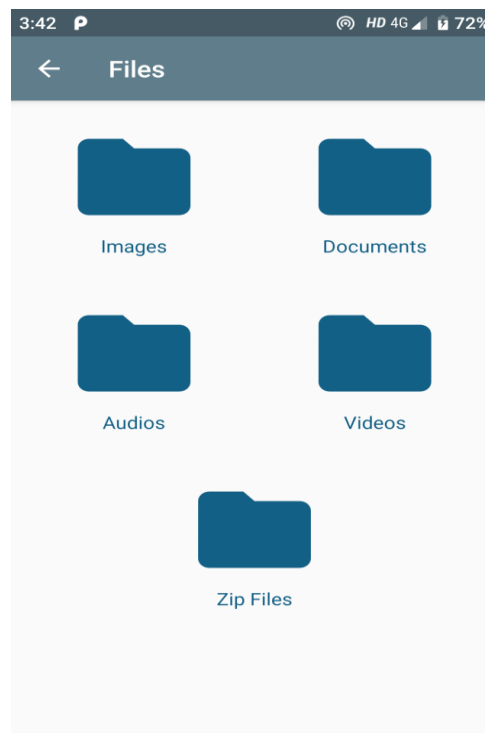


Fig 5.14: Files Activity

**Screen-12:** If we click on images we get a activity like this. Now we can add images here.

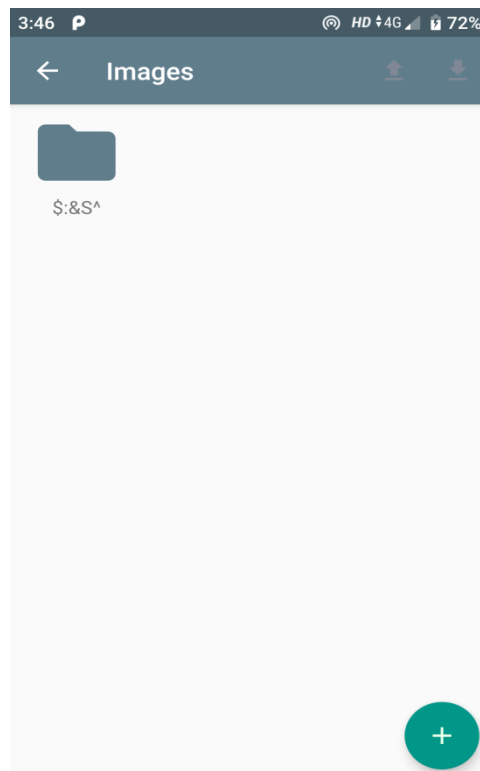


Fig 5.15: Images Activity

**Screen-13:** We can add videos if we click on videos by clicking floating action button.

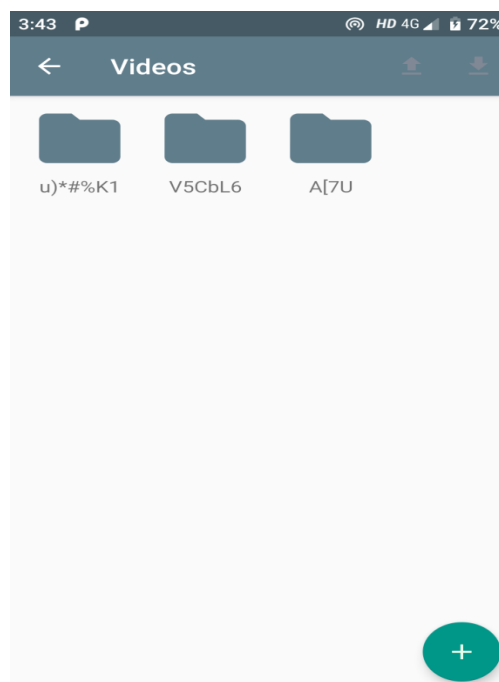


Fig 5.16: Videos Activity

**Screen-14:** If we click audio , we can add any file of audios here.

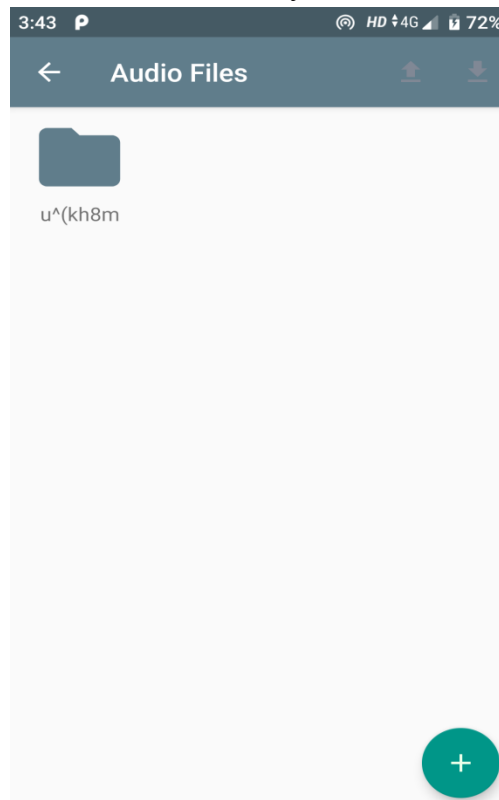


Fig 5.17:Audio Activity

**Screen-15:** In documents we can add different formats of documents like .pdf, .doc.

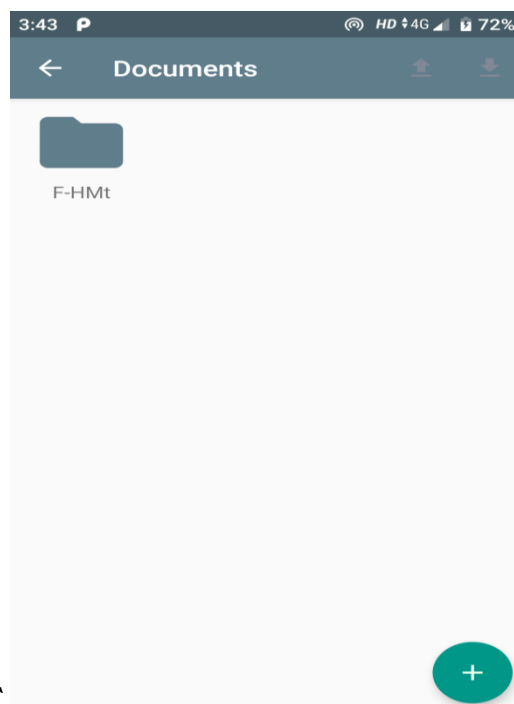


Fig 5.18:Documents Activity

**Screen-16:** If we click zip files we can add zip files and encrypt it.

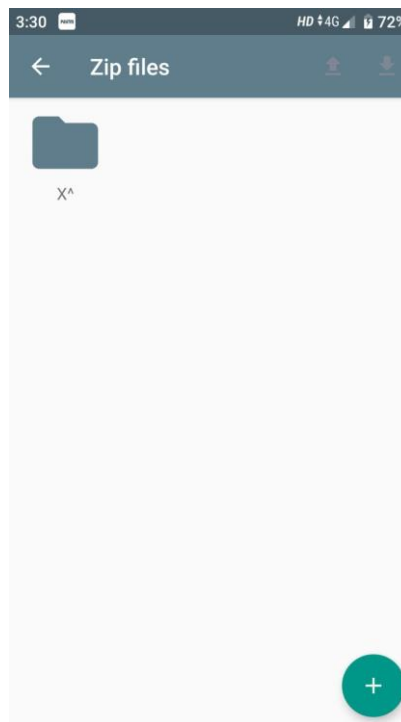


Fig 5.19: Zip files after adding zip file.

**Screen-17:** Let us demonstrate with a simple example. This is contacts activity and it contains upload and download activity. click on upload icon.

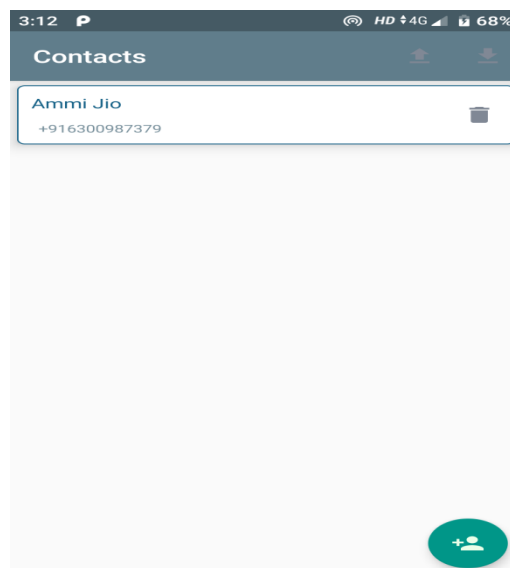


Fig 5.20: Uploading to cloud

**Screen-18:** In another mobile we need to install this application and login to the application. After login click on contacts and click on download icon and after clicking this option , here we can see the uploaded contact.

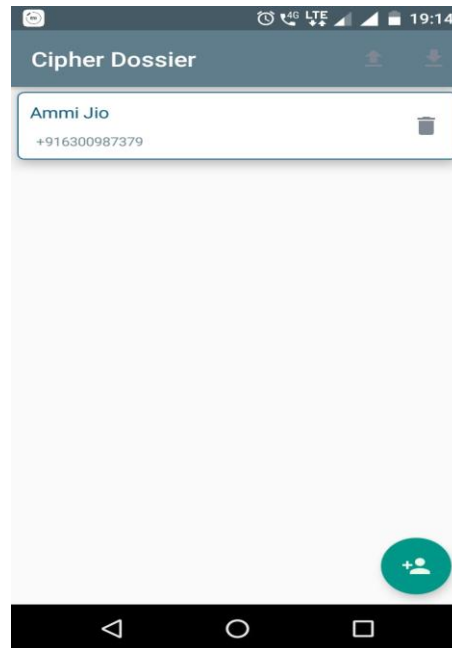


Fig 5.21:Contacts activity in another mobile

In this way we upload and download for all the slots provided in the application.

## 5.4 Summary

Implementation and results gives the explanation of key functions cipher dossier application with output screens and database tables used to design this application.



## CHAPTER-6

### TESTING AND VALIDATION

#### 6.1 Introduction

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and /or a finished product.

It is the process of exercising software with the intent of ensuring that the software system meets its requirement and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

#### 6.2 Design of Test cases and Scenarios:

The chapter which is presented below deals with the various tests that have been made to the developed software so as to detect the failures it may have. Along this chapter there will be carried out of test: **unit tests, integration tests and Espresso testing.**

##### 6.2.1. Unit Testing

Unit tests are the fundamental tests in your app testing strategy. By creating and running unit tests against your code, you can easily verify that the logic of individual units is correct. Running unit tests after every build helps you to quickly catch and fix software regressions introduced by code changes to your app.

A unit test generally exercises the functionality of the smallest possible unit of code (which could be a method, class, or component) in a repeatable way. You should build unit tests when you need to verify the logic of specific code in your app.

Table 6.1: Unit testing modules

S.NO	SCENARIOS	EXPECTED RESULT	ACTUAL RESULT	STATUS
1	Install cipherdossier.apk file on Android phone	Installation Successful	Installation Successful	Success
2	Check whether UI Is Displaying On screen	Display UI	Display UI	Success
3	Register/Sign In to the application	Sign In Successful	Sign In Successful	Success
4	Check whether home screen is visible or not.	Visibility of home screen is successful	Visibility of home screen is successful	Success
5	Selection of any slot like contacts or messages or notes or files.	Displaying the activity of respected slot is successful	Displaying the activity of respected slot is successful	Success

<b>6</b>	Adding the data like contacts or messages or notes of files (images, videos, documents, audios, zip files) into application.	Adding of data is successful	Adding of data is successful	Success
<b>7</b>	Upload the data to cloud	Uploaded	Uploaded	Success
<b>8</b>	Opening the application in another mobile and login and check the data uploaded.	Displaying the uploaded data	Displaying the uploaded data	Success

### 6.2.2 Integration Testing

Integration testing ensures that software and subsystems work together a whole. It tests the interface of all the modules to make sure that the modules have properly when integrated together.

### 6.2.3 Espresso

Espresso is a tool used for doing unit testing. Espresso is used to write concise, beautiful and reliable Android User Interface test. The core API is small, predictable, and easy to learn and yet remains open for customization. Espresso tests state expectations, interactions, and assertions clearly without the distraction of boilerplate content, custom infrastructure, or messy implementation details getting in

the way. Espresso tests run optimally fast! It lets you leave your waits, syncs, sleeps, and polls behind while it manipulates and asserts on the application UI when it is at rest.

Espresso is targeted at developers, who believe that automated testing is an integral part of the development lifecycle. While it can be used for black-box testing, Espresso's full power is unlocked by those who are familiar with the codebase under test. Espresso testing is applicable from 3.1.3 version of android studio. In Android, there are two types of tests:

- JUnit Tests
- Android instrumented unit test.

Espresso testing framework is usually used to automate UI testing with the help of Android JUnitRunner test runner. Make sure to add the following dependencies from the Android Testing Support Library in your **build.gradle** (although Android Studio has already included them for us).

```

        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
    }

    dependencies {
        androidTestImplementation 'androidx.test:runner:1.1.0'
        androidTestImplementation 'androidx.test.espresso:espresso-core:3.1.0'
    }

```

Fig 6.1: Dependencies for executing the espresso testing

We also included the instrumentation runner `AndroidJUnitRunner`:

*An Instrumentation that runs JUnit3 and JUnit4 tests against an Android package (application).*

Note that `Instrumentation` is simply a base class for implementing application instrumentation code.

## Getting Started

In order to test a UI create a new test class in the Location  
module-name/src/androidTest/java/

The instrumentation runner will process each test class and inspect its annotations. It will determine which *class* runner is set with `@RunWith`, initialize

it, and use it to run the tests in that class. In Android's case, the `AndroidJUnitRunner` explicitly checks if the `AndroidJUnit4` class runner is set to allow passing configuration parameters to it.

There are 6 types of annotations that can be applied to the methods used inside the test class, which are `@Test`, `@Before`, `@BeforeClass`, `@After`, `@AfterClass`, `@Rule`.

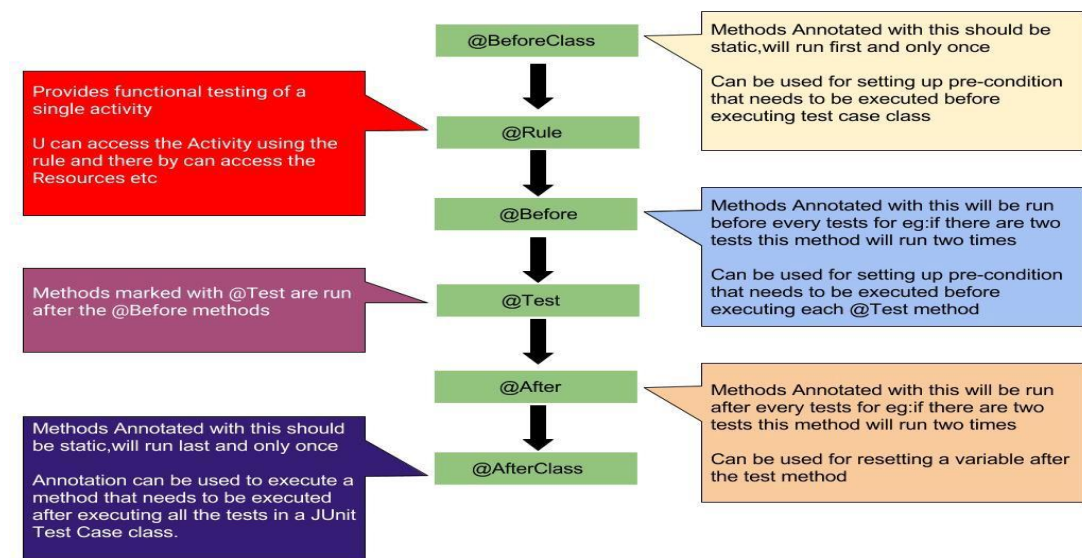


Fig 6.2: Annotations used in testing classes

Important things to note is that

- The activity will be launched using the `@Rule` before test code begins
- By default the rule will be initialized and the activity will be launched (`onCreate`, `onStart`, `onResume`) before running every `@Before` method
- Activity will be Destroyed (`onPause`, `onStop`, `onDestroy`) after running the `@After` method which in turn is called after every `@Test` Method
- The activity's launch can be postponed by setting the `launchActivity` to false in the constructor of `ActivityTestRule` in that case you will have to manually launch the activity before the tests.

The espresso test of a view contains

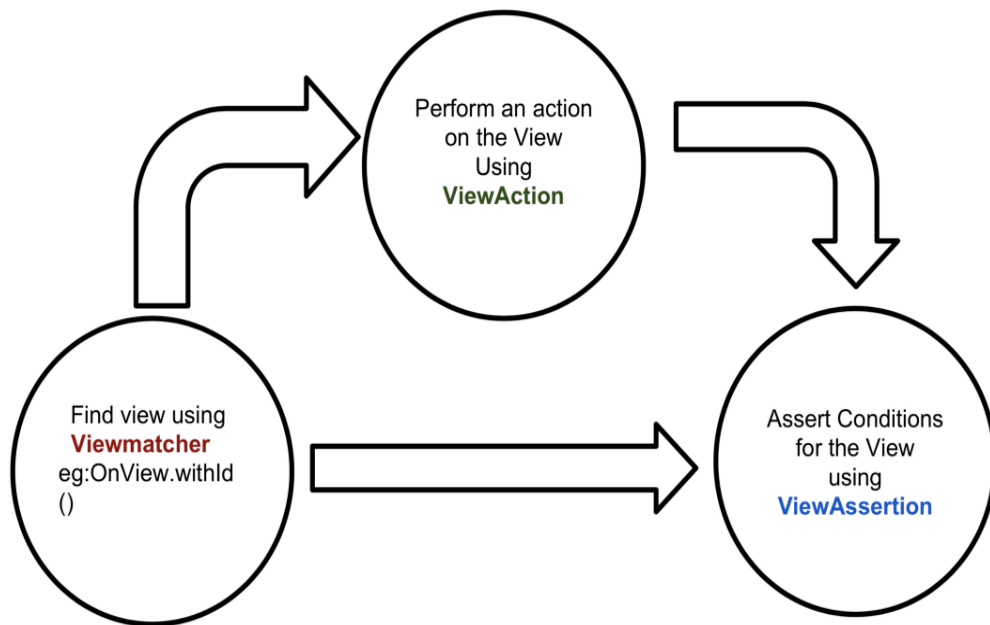


Fig 6.3: Steps involved in Espresso testing

### How to run the application on an android phone

- After running the application on the system once, an .apk file is generated automatically in the workplace.
- We now copy the .apk file from /bin on the computer to the phone and install it.
- After installing we run the application in the same way as we run it on a virtual machine.

### 6.3 Validation of test cases

At the culmination of integration testing the software is complete as a package and the interfacing errors have been uncovered and fixed, final tests - validation testing may begin. Validation tests succeed when the software performs exactly in the manner as expected by the user.

Following are the two cases which are being tested.

Case-1:Validates the email address whether it is registered or not

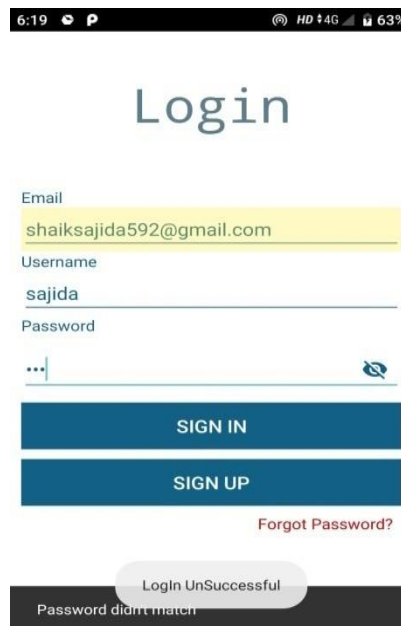


Fig 6.4: Validating the email address

Case -2:If internet connection is not available then displays an alert to turn on the mobile data.

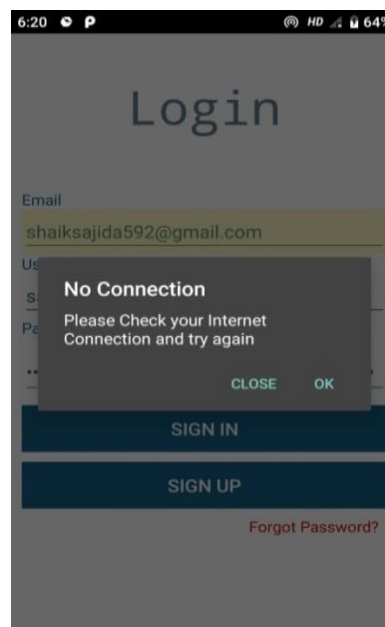


Fig 6.5: Showing alert Message

## 6.4 Summary

Testing and validation gives the correctness of the output of the application by using some testing techniques.

## CONCLUSION

The application mainly concentrates on mobile phones and efficiently handles the situation when a mobile is lost and when there is no way to retrieve the data. In this application, the data like contacts, messages, notes, images and videos can be encrypted using a key generated by the user and this key is used to encrypt the contents that are chosen by the user to keep them confidential. The user has to login into the application to make use of the application. The data in this application is stored on the cloud storage and stored in an encrypted form using the key that is generated by the user. Even if the mobile is lost, no one can access the data as it will be encrypted by the key that is known to the user only. The data is decrypted only by the key that is used to encrypt it and if the key is changed, the data cannot be retrieved in its original form. And if the user wants to access the data he can login into the application with their credentials from any device using the key that is used to encrypt the data.



## BIBLIOGRAPHY

- [1]Ako Muhamad Abdullah. (2017). Advanced Encryption Standard (AES) Algorithm to Encrypt and Decrypt Data.
- [2]Vichare, A., Jose, T., Tiwari, J., & Yadav, U. (2017). Data security using authenticated encryption and decryption algorithm for Android phones, 2017 International Conference on Computing, Communication and Automation (ICCCA). doi:10.1109/cca.2017.8229903.
- [3] Chen, Y., & Ku, W.-S. (2009). Self-Encryption Scheme for Data Security in Mobile Devices. 2009 6th IEEE Consumer Communications and Networking Conference. doi:10.1109/ccnc.2009.4784733
- [4] Mitesh Parmar, Summedh Kharat, Nilesh Palve, Chetan deokate, Prof. Yogesh shahare, Mobile Self Encryption Techniques, International Journal for Research in Engineering Application & Management (IJREAM), 12 March 2018.
- [5] <https://firebase.google.com/>