Software Design Document (SDD) Template

Software design is a process by which the software requirements are translated into a representation of software components, interfaces, and data necessary for the implementation phase. The SDD shows how the software system will be structured to satisfy the requirements. It is the primary reference for code development and, therefore, it must contain all the information required by a programmer to write code. The SDD is performed in two stages. The first is a preliminary design in which the overall system architecture and data architecture is defined. In the second stage, i.e. the detailed design stage, more detailed data structures are defined and algorithms are developed for the defined architecture.

This template is an annotated outline for a software design document adapted from the IEEE Recommended Practice for Software Design Descriptions. The IEEE Recommended Practice for Software Design Descriptions have been reduced in order to simplify this assignment while still retaining the main components and providing a general idea of a project definition report. For your own information, please refer to IEEE Std 1016-1998 [1]for the full IEEE Recommended Practice for Software Design Descriptions.

[1] http://www.cs.concordia.ca/~ormandj/comp354/2003/Project/ieee-SDD.pdf

The Only Member Team

Encryptor

Software Design Document

Name: Omer Edut
Lab Section:
Workstation:

Date: 03/13/2020

Table of Contents

1. Introduction

1.1 Purpose

This SDD document describes the architecture and system design of *Encryptor*, and is intended for program developers to define the required functions and technological guidelines. Also, to the software testers as the basis for designing the test scripts.

1.2 Scope

Encryptor is a mobile app designed to encrypt or decrypt an image with sensitive information / content within another image so as not to arouse suspicion or interest among other stakeholders.

1.3 Overview

This SDD document provides a general explanation of the *Encryptor* product, its purpose and for who it is intended. This explanation can be found in Section 1 of this document. In section 2 you will find an overview of the system. Section 3 of this SDD document includes the architecture of this product in high-level prespective. In addition, this SDD document contains the data design of this product, this part will be found in section 4. In section 5 you will find the component design of this product in a clear and detailed manner. In section 6 you will find the functionallity of this product from the customer prespective.

1.4 Reference Material

This section is not relevant in this SDD document.

1.5 Definitions and Acronyms

This section is not relevant in this SDD document.

2. System Overview

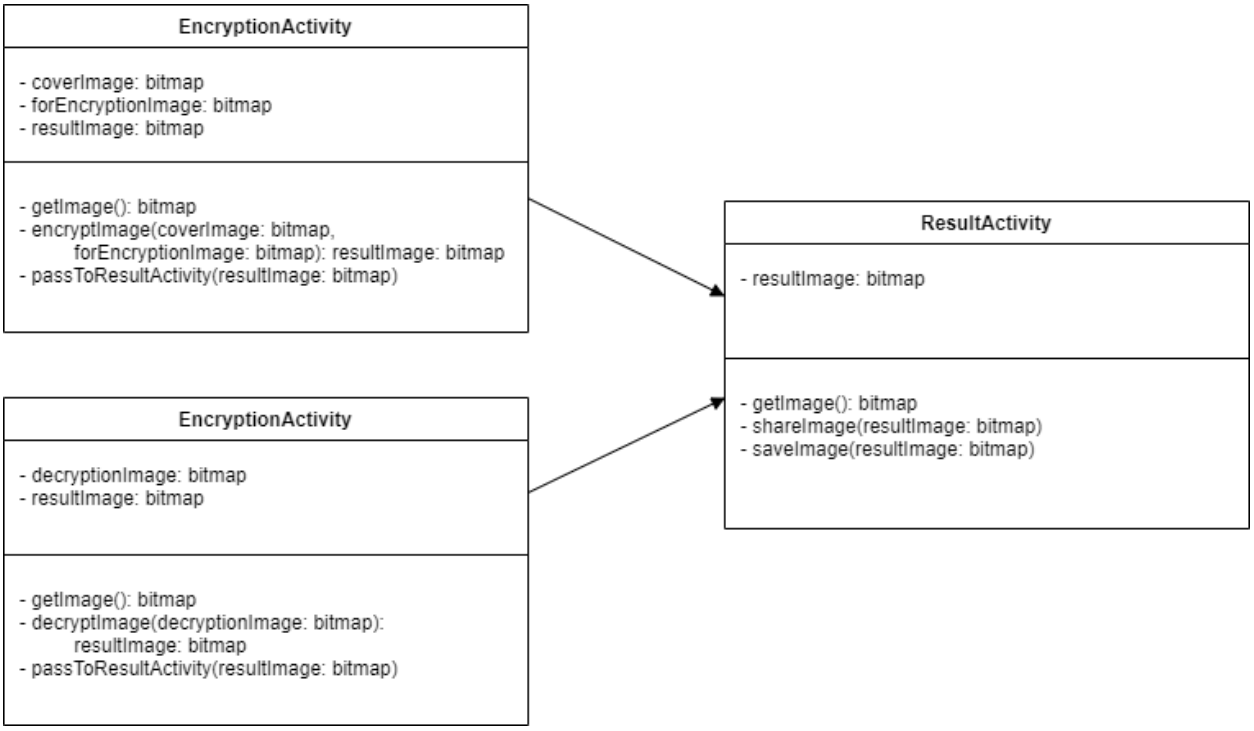
Encryptor is a mobile application that is designed in MVVM design pattern that allows you to create an encrypted image within another image, or decrypt an encrypted image within another image.
When encrypting an image, the app will require two images (cover image and encryption image).
After entering the data the app will encrypt the image and display the result.
When decoding an image, the application will require two images (a cover image and the encrypted image) and return the decoding result.
The app will allow the result to be exported to an image file, or transferred to another messaging app.

3. System Architecture

3.1 Architectural Design

Encryptor app has three main modules, EcryptionActivity, DecryptionActivity and ResultActivity (Figure 1).
EncryptionActivity - Its role is to recieve from the user two images (cover image, encryption image), send the images to the encryption model using the View Model, and get the result.
DecryptionActivity - Its role is to recieve from the user one image containing an encrypted image, send it to the decoding model, using the View Model, and get the result.
ResultActivity - Its role is to receive and display the result image from the modules mentioned above, this module allows you to perform further actions that will be detailed below.

Figure 1: Main Modules Diagrams

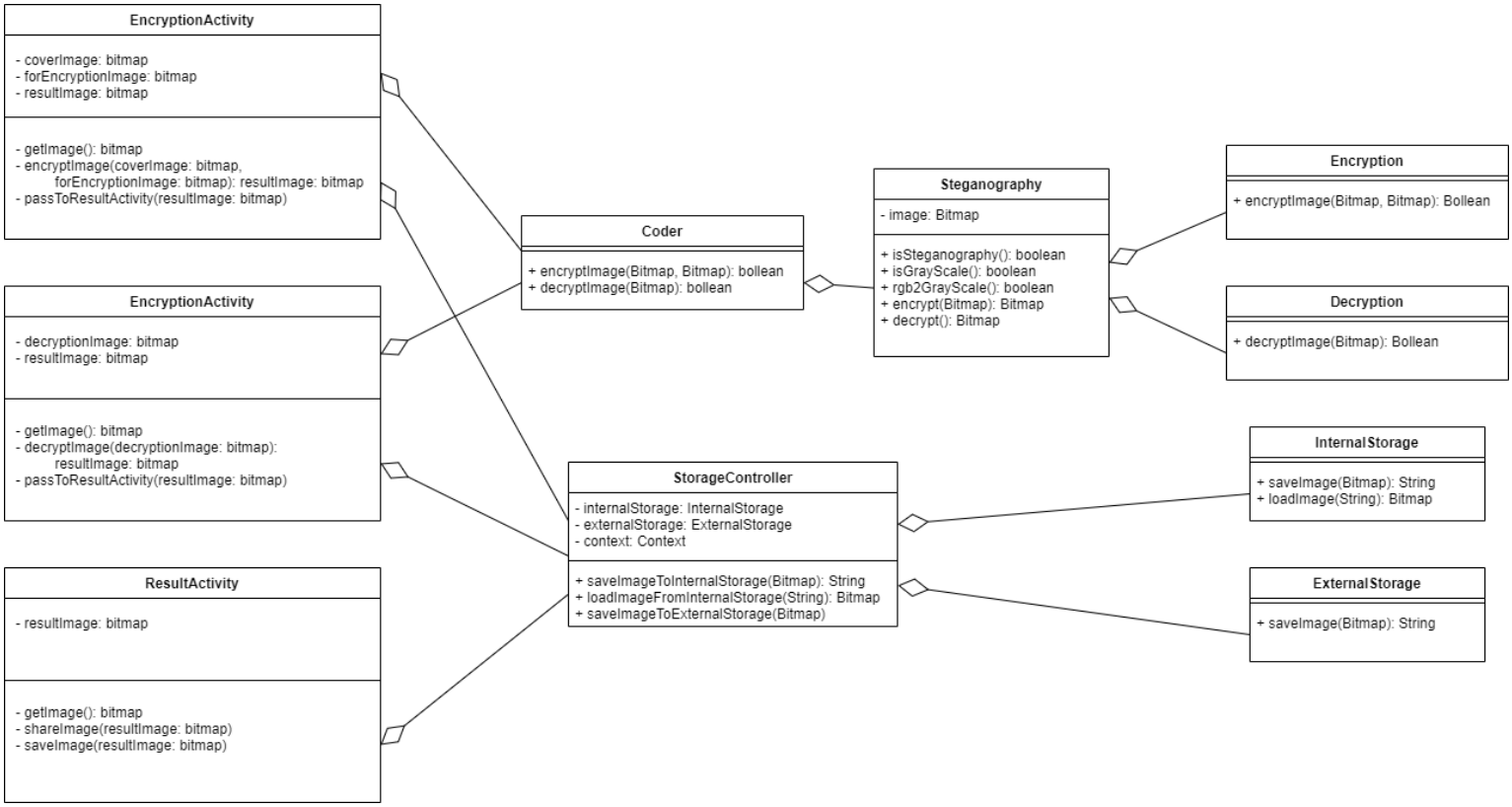


3.2 Decomposition Description

This section presents the decomposition description of the app's screens and their actions by object-oriented description.
Attached are the object diagrams (Figure 2) and sequence diagrams (Figures 3,4,5).

3.2.1 Object Diagrams

Figure 2: Object Diagram



3.2.2 Sequence Diagrams

Figure 3: EncryptionActivity sequence diagram

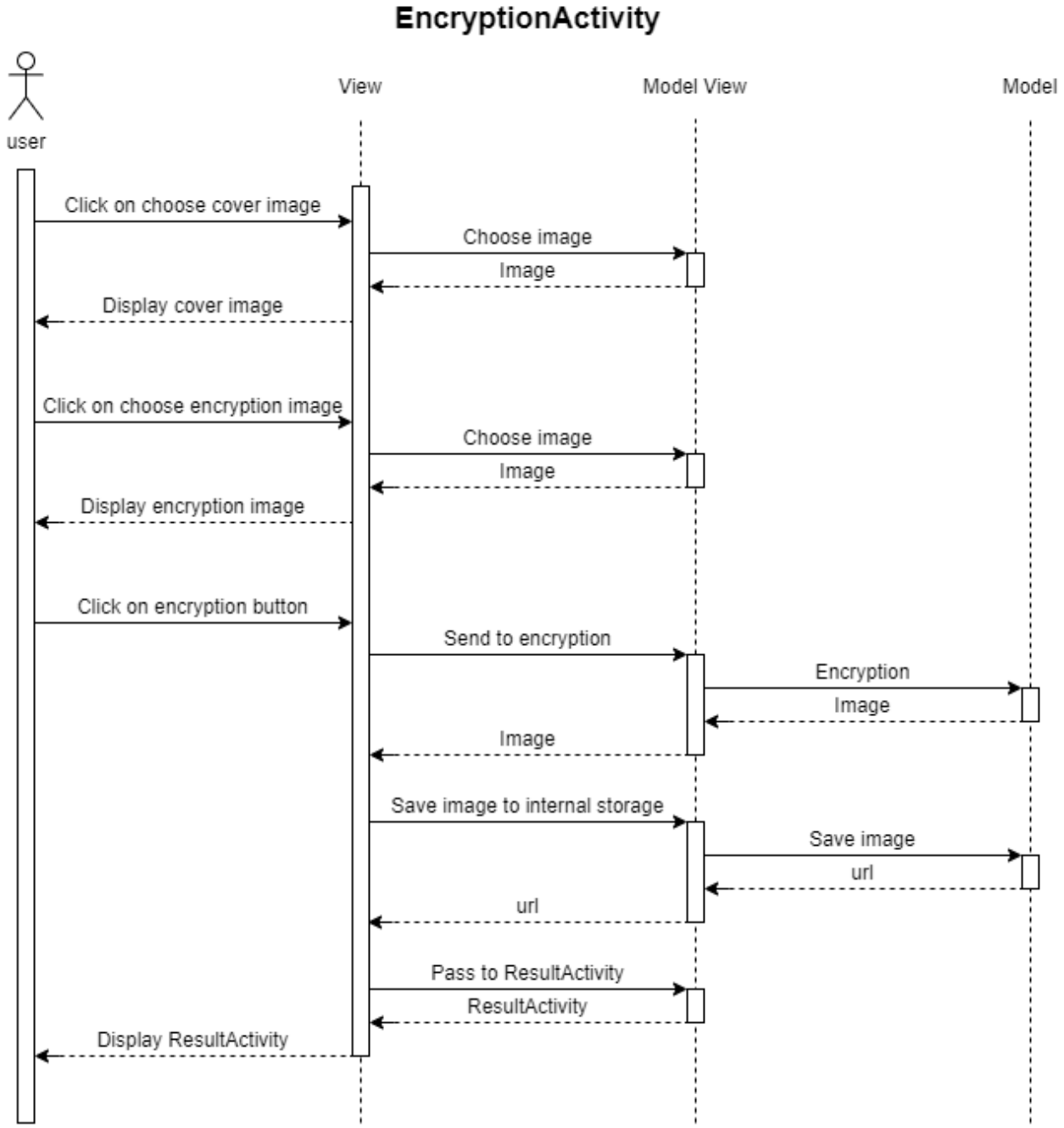


Figure 4: DecryptionActivity sequence diagram

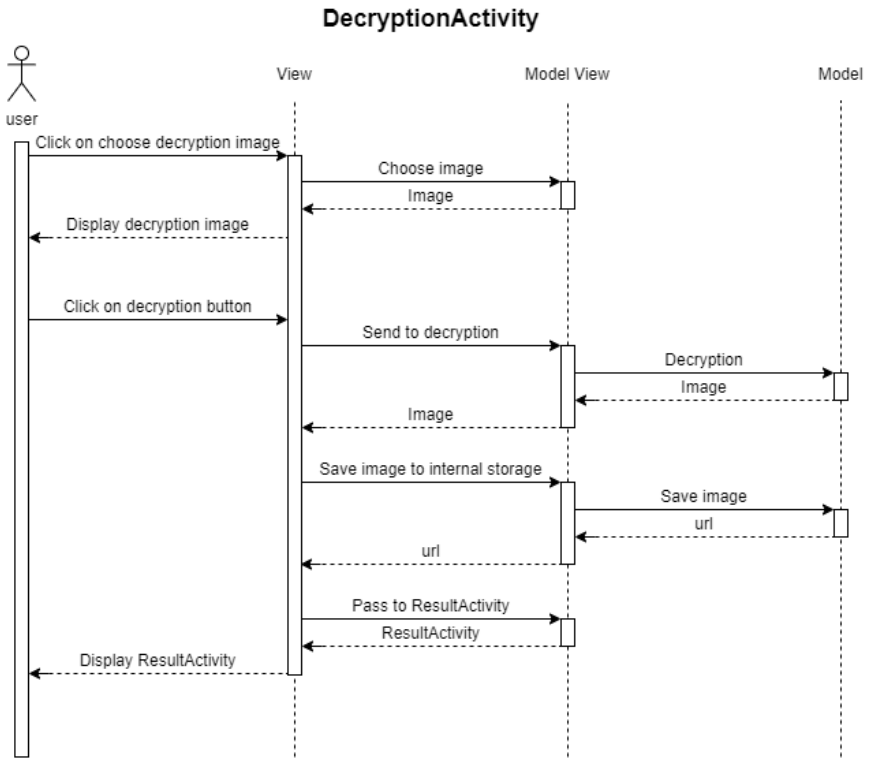
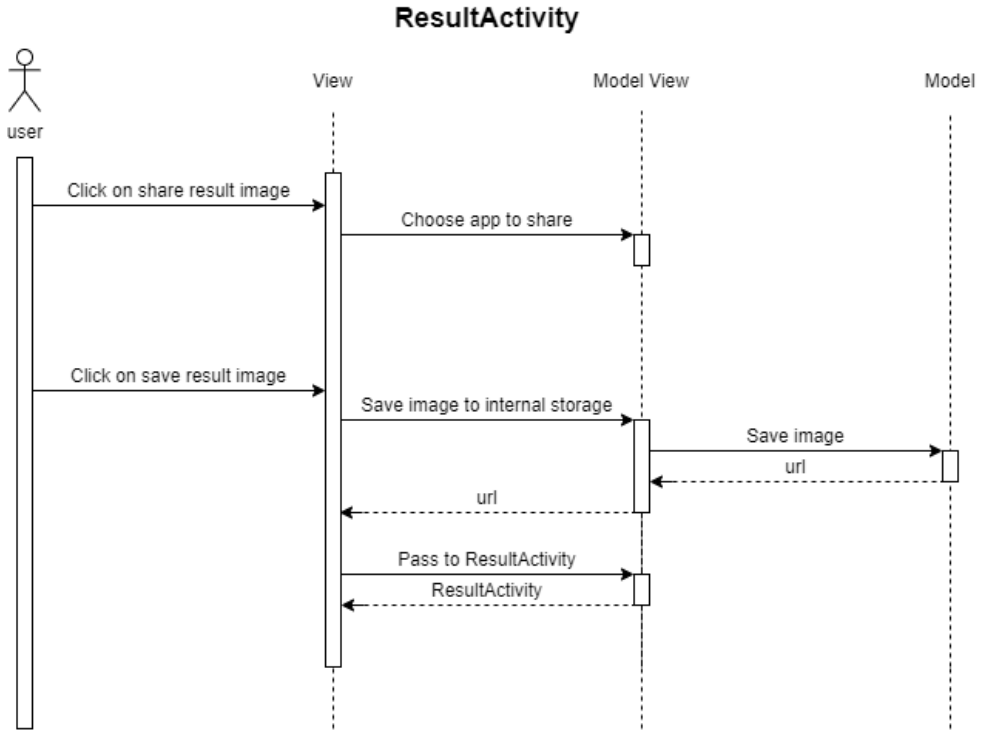


Figure 5: ResultActivity sequence diagram



3.3 Design Rational

In Android apps in general and in *Encryptor* in particular, the most convenient and intuitive way to design the project is in VMMV design pattern. The reason is that in a structured way, there are files that are responsible for the user's view, making it easy to separate the data structure, allowing control and order of the project. This structure also allows easy and simple modification of the code, data structure testing and user interface testing separately and right maintenance.

4. Change Management Process

4.1 Data Description

At this point, the system does not have databases, but when you switch between the encryption / decryption screen to the result screen the result image is saved in the app's internal storage to transfer it to the result screen (these are some of the Android system restrictions). If the result is suitable for the user and he wants to save it, the same image will be stored in the external storage of the device.

4.2 Data Dictionary

DecryptionImage is a Bitmap type object, it represent the bitmap image for decryption.
EncryptionImage is a Bitmap type object, it represent the bitmap image for encryption.
CoverImage is a Bitmap type object, it represent the bitmap image for cover the encryption image.
Steganography object it is a data structure used to carry out operations for image encryption or decryption.
The object requires Bitmap format and if it's a grayscale image using the method *isGrayScale()*, it's possible convert a grayscale image by the method *rgb2grayscale()*, and also send it to encrypt using the method *encrypt()* or decrypt an image using the method *decrypt()*.

5. Component Design

```
Begin saveImage(img: Bitmap)
   Create a fileName
   Create a file by fileName & StorageDirectory
   Write img to file
   Return filename
End saveImage

Begin loadImage(fileName: String)
   Create a filePath by fileName & InternalStorageDirectory
   img  = decode file by filePath
   Return img
```

```
End loadImage

Begin shareImage(fileName: String)
   Create a file by fileName & StorageDirectory
   Uri = get uri from file
   Create shareIntent
   shareIntent.setImageParamsForShare
   startActivity with shareIntent
End shareImage


Begin passToResultActivity(filePath: String)
   Create resultIntent
   resultIntent.setString → filePath
   startActivity with resultIntent
End passToResultActivity


Begin encryption
    row = 2
    while (row ≤ n-2) and (the secret image is not finished)
       col = 2 + (row MOD 2)
       while col ≤ m-2
          x = stegoC(row-1,col) ⊕ stegoC(row+1,col) ⊕ stegoC(row,col-1) ⊕
                stegoC(row,col+1))
         if x ≤ α
            numLSBs = 1
         else
            numLSBs = ⌈x/2⌉
         endif
        replace LSBs of stegoC(row,col) with the next numLSBs bits from the secret
                image
         col = col + 2
      endwhile
      row = row + 1
   endwhile
end encryption
```

6. Supporting Information

6.1 Overview of User Interface

When opening the app, the user can choose the action he wants to do, encryption or decryption (Figure 6).

Clicking on one of the actions will take the user to the appropriate screen.
In the encryption screen, the user needs to enter two images, a cover image and an encryption image. The user will then be able to click on the encryption execution button. The user may also decide that he dosen't want to do this action and can return to the previous screen (Figure 7).
In the decryption screen, the user has to enter one image, and then he can click on the decryption button. The user may also decide that he dosen't want to do this and return to the previous screen (Figure 9).
After performing these actions (Figure 10), in case the action fails, an appropriate message will be displayed to the user and in case the action is successful the user will be taken to the result screen (Figure 11). On this screen the user will see the result image and can save it to the device or send it to another application. The user can also decide that he is not interested in the result and return to the previous screen.

6.2 Screen Images

Figure 6: Choose action

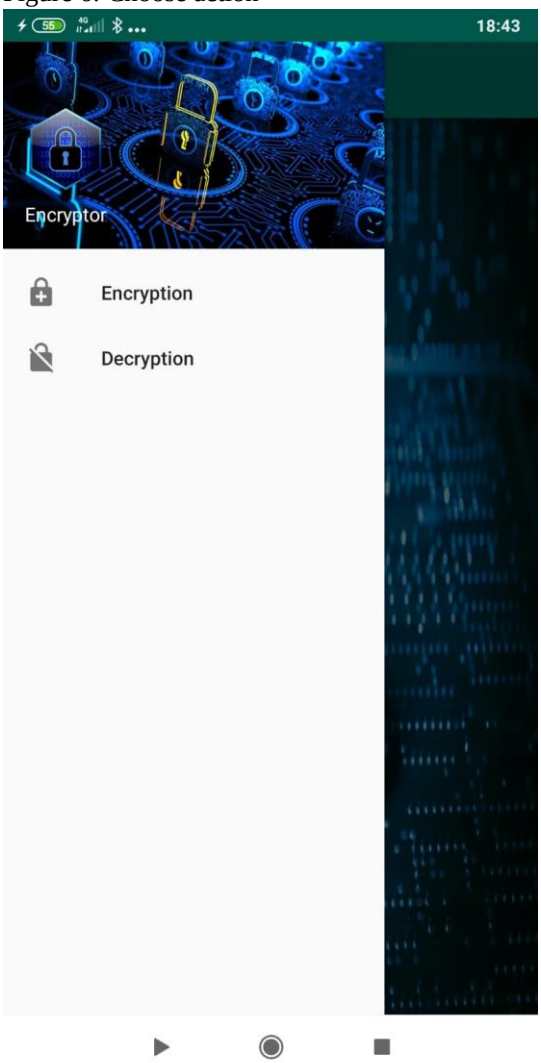


Figure 7: Encryption screen

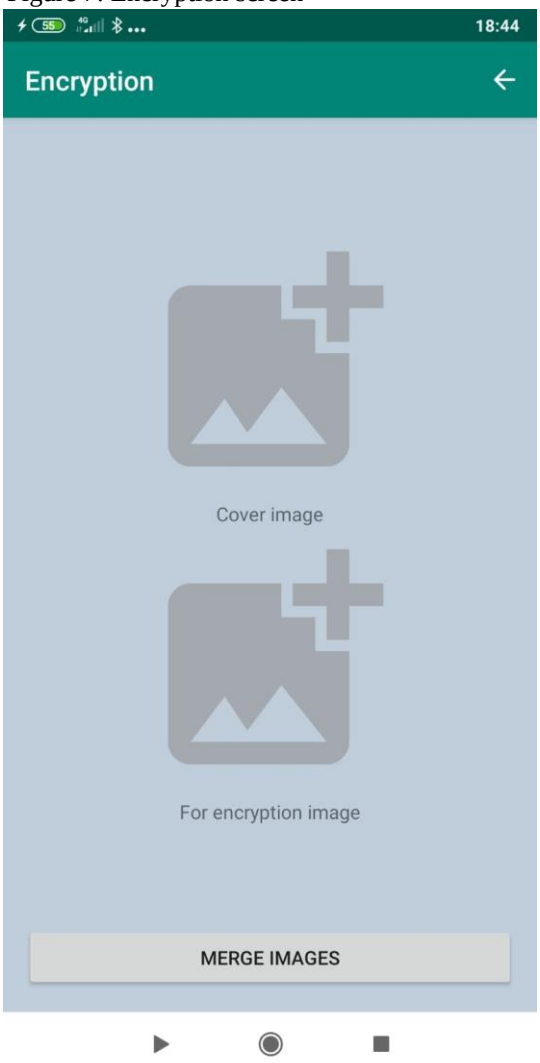


Figure 8: Choose image

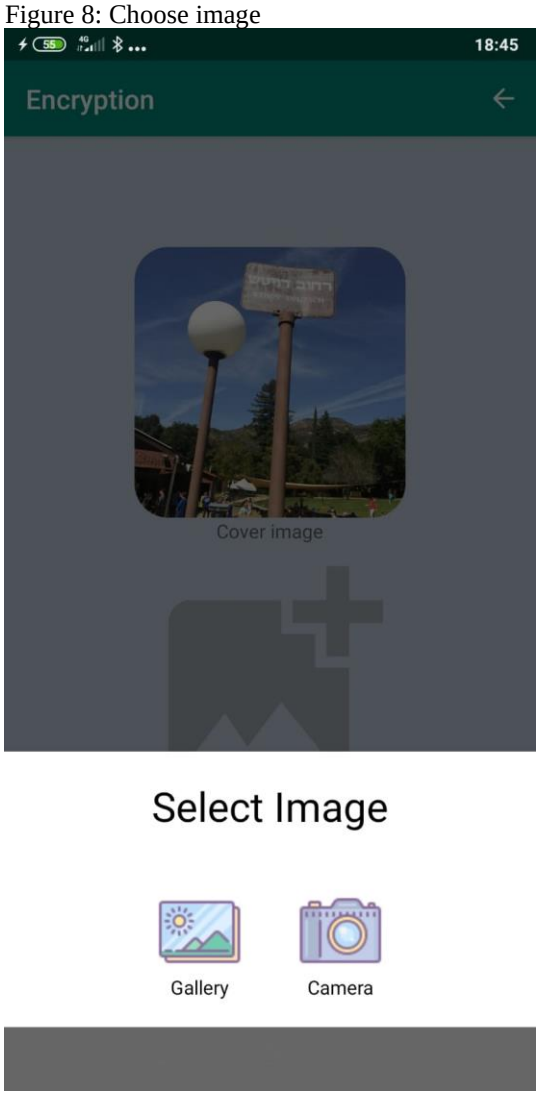


Figure 9: Decryption screen

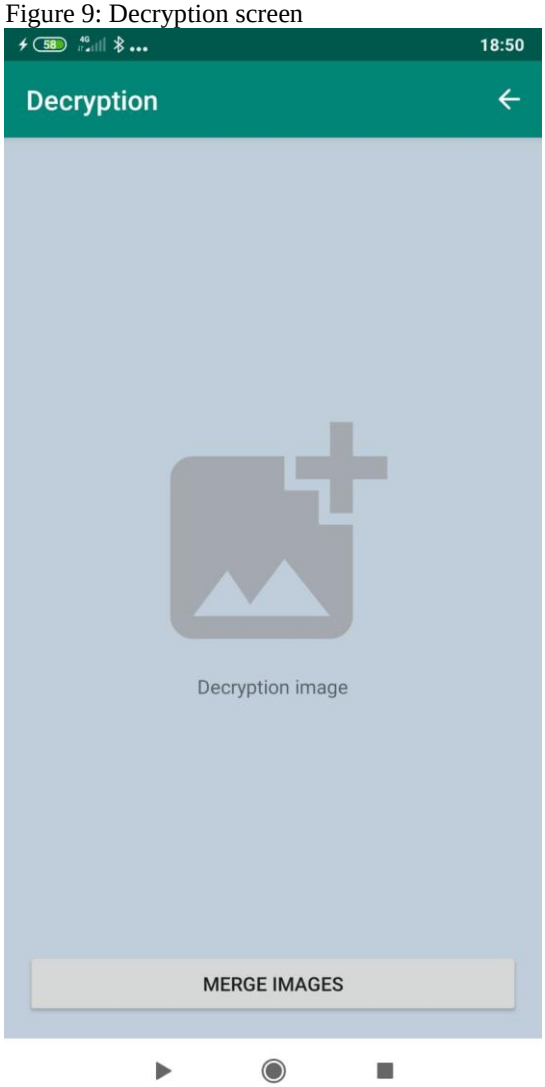


Figure 10: Two images have been chosen

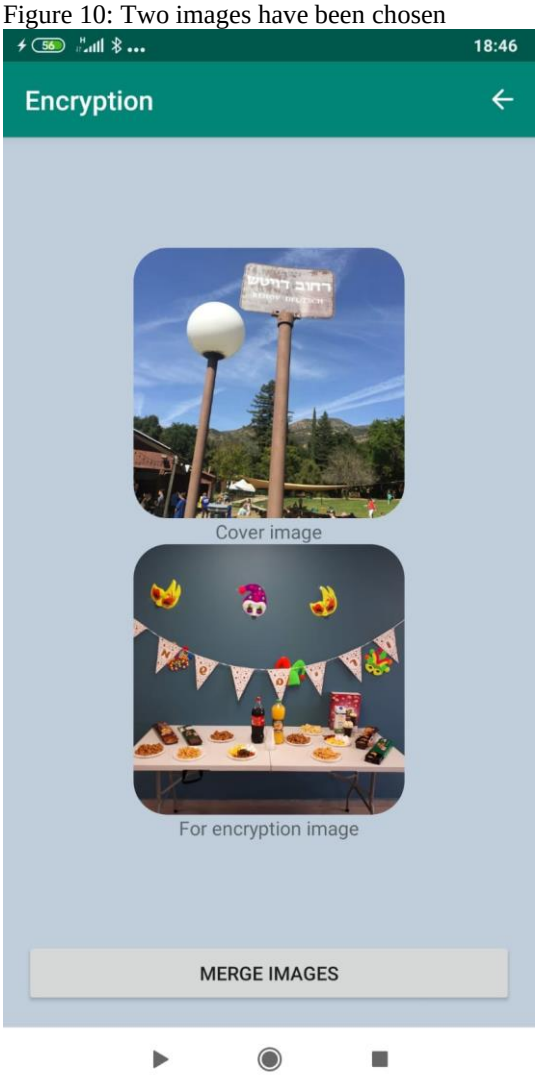


Figure 11: Result of encryption

Figure 12: Share image

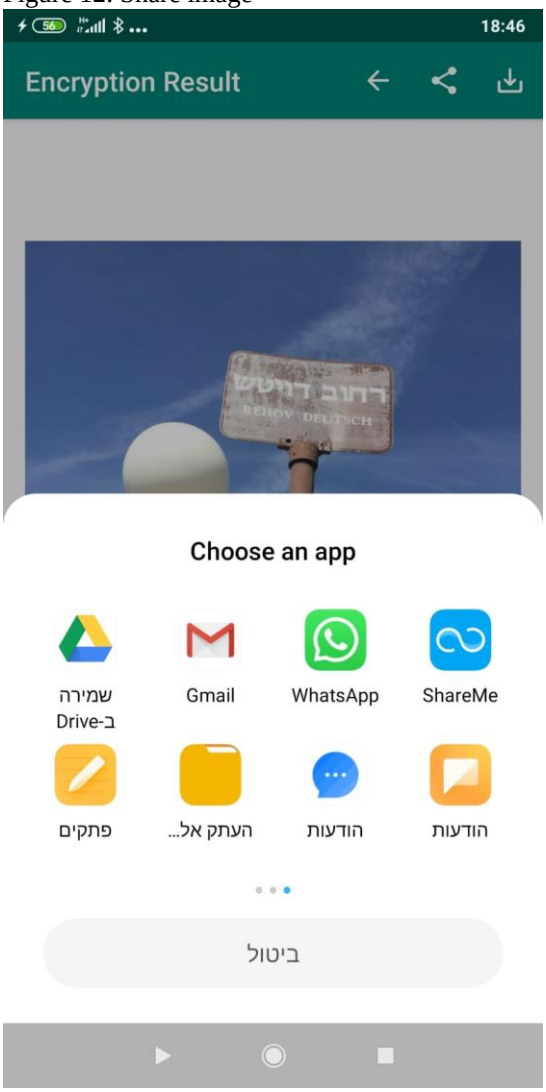
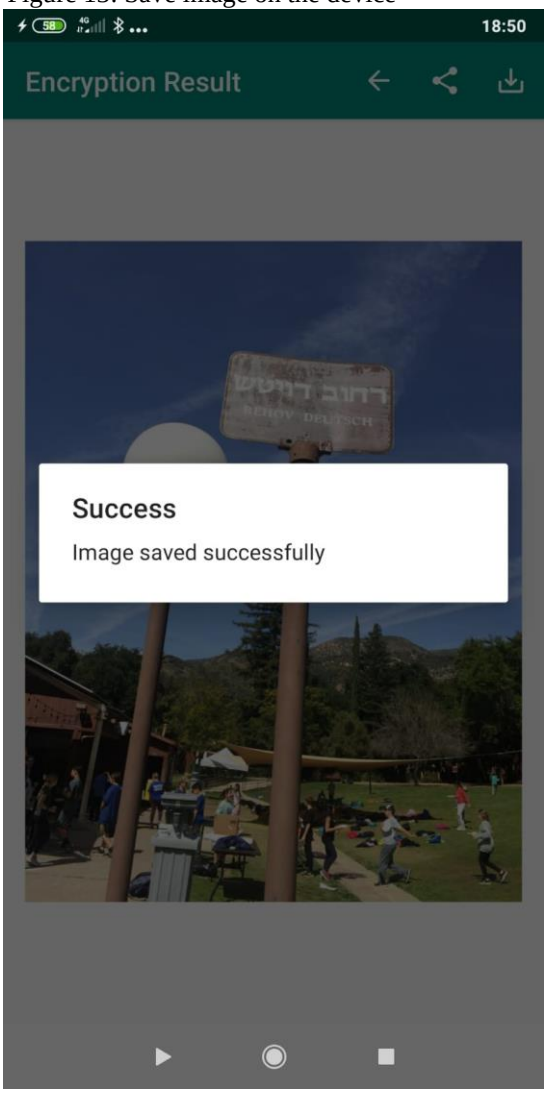


Figure 13: Save image on the device



6.3 Screen Objects and Actions

6.3.1 Choose an Image

When clicking an image, the user can choose how to import the image into the app (from the gallery or from the camera of the device). After selecting the image, the image is displayed to the user on the app screen. By further clicking on the image, the user can delete the image or select another image (Figure 8).

6.3.2 Encryption/Decryption Button

When clicking the encryption / decryption button, the operation will only be performed if all the required data has been entered (for encryption - enter cover image and encryption

image, for decryption - enter decryption image), otherwise the user will be notified that the process cannot be continued.
If all the required data is entered, the encryption / decryption process will begin, if the process fails, a suitable message is displayed to the user, if the process succeeds the user will be taken to the result screen, as explained in section 6.1

6.3.3 Share Button

When clicking on the share button, the user will be able to select from the list of apps that are on their device, to which app they want to transfer the result image (Figure 12).

6.3.4 Save Button

When clicking the save button, the image will be saved on the app folder in the external storage of the device. If the saving fails, an appropriate message will be displayed to the user (Figure 13).

7. Requirements Matrix

Figure 14: Explanation for the requirements from the SRS document

Id	Requirement	Explanation	priority
1	Insert cover image	Add image in EncryptionActivity	1
2	Cancel cover image	Clicking on exist image in EncryptionActivity	1
3	Insert encryption image	Add image in EncryptionActivity	1
4	Cancel encryption image	Clicking on exist image in EncryptionActivity	1
5	Insert parameters ecryption		1
6	Encryption algorithm	Clicking on encryption button on EncryptionActivity	1
7	Insert decrypt image	Add image in DecryptionActivity	2
8	Cancel decrypt image	Clicking on exist image in DecryptionActivity	2
9	Decryption algorithm	Clicking on encryption button on DecryptionActivity	2
10	Show result image	Pass to ResultActivity	1
11	Export/Save result image	Clicking on save result button in ResultActivity	1
12	Send result to some messaging app	Clicking on share button in ResultActivity	3

8. Appendices

This section is not relevant in this SDD document.