Report

# Project Report

CSE299: Junior Design

**Course**: CSE299.06

**Semester**: Fall 2022

**Submitted to**: Dr. Nabeel Mohammed (NbM)

**Submitted by**:

| **Name** | **ID** |
|---|---|
| Md Sahadul Hasan Arian | 2011084042 |
| Nadman Ashraf Khan | 2012466642 |
| Tahmid Ashraf Khan | 2012711042 |
| Samira Ali | 2011423042 |

# Abstract

This paper aims to introduce the system features, designs, implementation details, and system evaluation results of the application **Drivecryptor** to a bigger audience. The paper's cornerstone is the systematic evaluation because this report is a part of an engineering course project that defines metrics, methods, details, and a summary of the assessment. The evaluation in this paper is divided into correctness, and usability evaluation, described and appropriately defined further in a later section. To conclude the system evaluation results, however, we must understand the system designs carefully that affect the evaluation's relevant metrics. Hence, the report also aims to establish a detailed analysis of the system by defining the project details, use cases, UI prototypes, Database details, Third Party Component libraries used, and high-level component interactions of the system as well as Process and Information flow. Alongside the correctness evaluation, another team completed the usability evaluation to find a performance metric for the system's usability. In the end, the evaluation results indicate that the system is remarkably stable under rigorous and exhaustive test cases; for further details, refer to the summary section of the system evaluation section.

# Contents

# 1. Introduction

Drivecryptor is a mobile application that facilitates client side encryption and decryption for users' Google Drive files. It provides a high degree of privacy and security for the user's cloud files by requiring face-based biometric authentication for every user action.

The files are encrypted into the Google Drive space when they are uploaded to block unauthorized access. When the authorized user wants to preview the file contents, the system downloads the encrypted file and decrypts the contents for use. Also, if the user's phone falls into the hand of an unauthorized user, those files need further protection along with encryption/decryption means. As a result, face recognition is used to check if the user that wants to access proprietary files is the authorized user in the true sense.

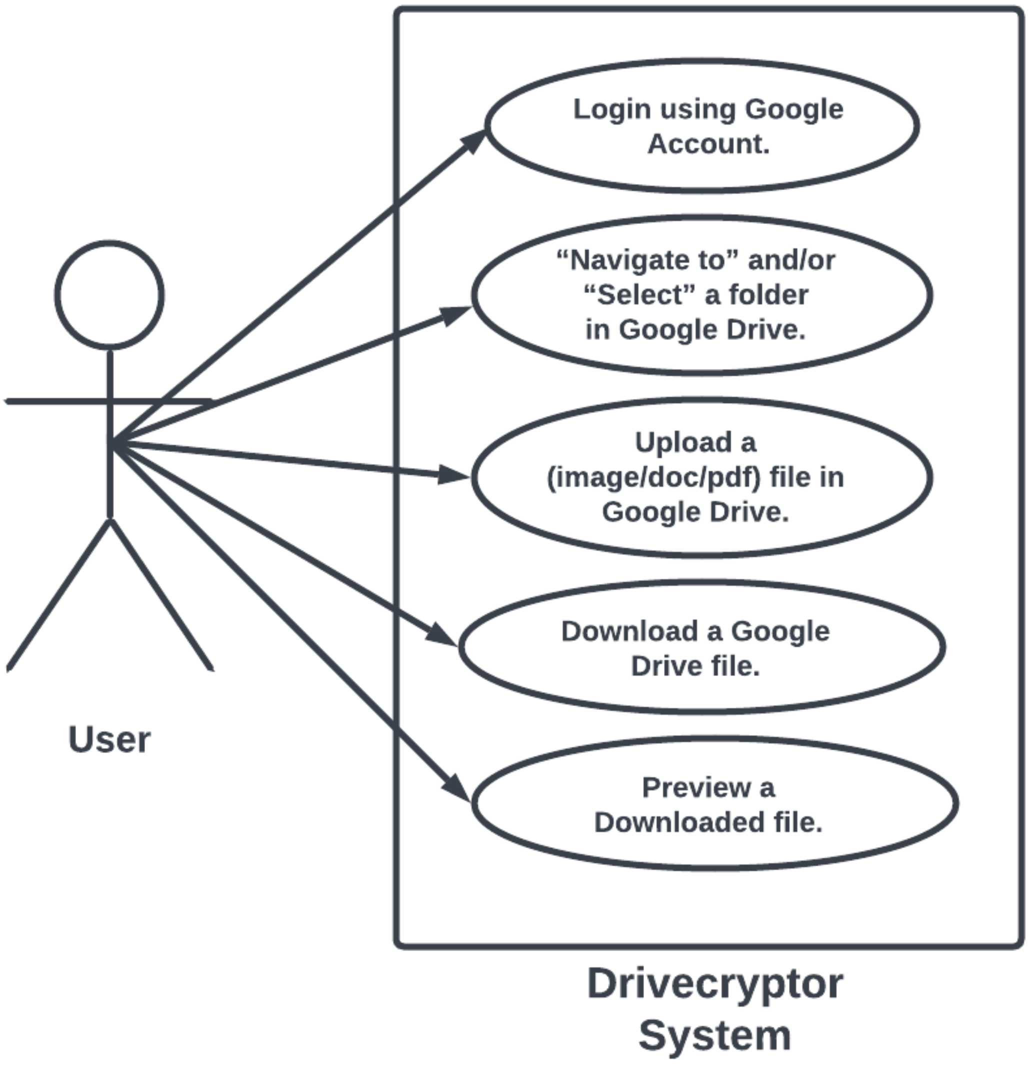
The paper/report is structured in such a way that it serves the purpose of both an SRS and an SDS alongside the evaluation section:

- A brief introduction is presented to the readers (current section). The core features are also highlighted.
- After the introduction, the system features section succeed. The use cases and their expanded forms are presented along with UI Design prototypes.
- The database design details, API documentation, data flow, and high-level system component interactions are presented.
- Then, the implementation details are written. Our team planned for the whole implementation to finish within five weeks. All five weeks' detailed work breakdown is presented.
- In the system evaluation, the report presents the relevant metrics, evaluation methods, and evaluation implementation details that supply enough details to repeat the assessment independently.

Finally, everything is concluded in the conclusion section with a summary of all results and designs.

# 2. System Features

## 2.1 Use Case Diagram



Login using Google Account.
“Navigate to” and/or “Select” a folder in Google Drive.
Upload a (image/doc/pdf) file in Google Drive.
Download a Google Drive file.
Preview a Downloaded file.
User
Drivecryptor System

Figure 1: Use Case Diagram

## 2.2 Expanded Use Cases

### 2.2.1 Use Case 1

| | |
|---|---|
| **Use Case** | Login using google account. |
| **Actors** | User |
| **Purpose** | Authenticate the user to the system using a google account. |
| **Overview** | The user decides to log into the software using the user interface. To do that, they click the ”Sign in with Google,” and the user can choose the email address they want to use to sign in to the system from the Google Play Prompt. When the user successfully logs into their Google account, they are redirected to the software's user interface, and a message is relayed. The system will notify the user if the logging event is a success or failure. Upon successful authentication, if the user is new to the system (i.e., they never logged into it), the system will prompt them to click their face picture. However, suppose the user has an existing account. In that case, the software will directly open the dashboard. The reference picture is cached during each login, and all the configurations are synced. |
| **Type** | Primary. |
| **Cross References** | Not Applicable. |

**Typical Course of Events**

| Actor Action | System Response |
|---|---|
| 1.This use case begins when the user chooses to log into the system. | |
| 2. The user clicks/touches the ”Login” button from the user interface. | |
| | 3. The system presents a prompt to the user via the user interface that asks for the Google Play's User Email. |
| 4. User Selects the Email Address. | |
| | 5. System Initiates login sequence. |
| | 6. After Completing the login sequence, user is redirected to Dashboard. |

**Alternative Courses**

- Section 5. If The user has a reference picture saved in AppData, then user is prompted to click a picture.

### 2.2.2 Use Case 2

| | |
|---|---|
| **Use Case** | ”Navigate to” and/or ”Select” a folder in Google Drive. |
| **Actors** | User |
| **Purpose** | The user navigates to the google drive space and selects the folder for possible destination for another Use Case (Use Case 3). |
| **Overview** | The user decides to navigate to ”My Drive” and view the file lists in Google Drive for all files and folders. They may select the folder for possible upload destinations for Use Case No 3. |
| **Type** | Primary. |
| **Cross References** | Use Case 3. |

**Typical Course of Events**

| **Actor Action** | **System Response** |
|---|---|
| 1. This use case begins when the user clicks ”My Drive” button in the dashboard user interface. | |
| | 2. The system prompts the user for face verification. |
| 3. The user aligns their face to the camera to verify the face. | |
| | 4. The system checks the face for verification against the reference picture and also runs anti-spoof mechanism. |
| | 5. Upon verification, the system shows all the files in the root folder and waits for further navigation from the user |
| 6. The user may select/navigate into the folders and choose that folder as possible destination for upload in Use Case 3. | |

### 2.2.3 Use Case 3

| | |
|---|---|
| **Use Case** | Upload a file in Google Drive. |
| **Actors** | User |
| **Purpose** | The user selects his file location and possible upload destination and clicks upload. |
| **Overview** | The user selects his file location (often may come from use case 2) and possible upload destination and clicks upload. The system runs encryption and uploads the encrypted file in Google Drive. |
| **Type** | Primary. |
| **Cross References** | Not Applicable. |

**Typical Course of Events**

| **Actor Action** | **System Response** |
|---|---|
| 1.This use case begins when the user clicks ”Upload File” button in the dashboard user interface. | |
| | 2. The system prompts the user for face verification. |
| 3. The user aligns their face to the camera to verify the face. | |
| | 4. The system checks the face for verification against the reference picture and also runs anti-spoof mechanism. |
| | 5. Upon verification, the system shows the upload interface. |
| 6. The user selects the file from their phone using the android file picker | |
| 7. The user selects the destination from either Navigating from ”My Drive” interface or clicking the ”Choose Destination” button. | |
| 8. The user clicks the upload button. | |
| | 9. The system starts encrypting file & Enters the task in Uploading Tasks List. |
| | 10. The system starts uploading the file. |
| | 11. Upon completing the upload, the system prompts a notification that the uploading task has completed and marks the file as uploaded in uploading task list. |

### 2.2.4 Use Case 4

| | |
|---|---|
| **Use Case** | Download a Google Drive file. |
| **Actors** | User |
| **Purpose** | The user navigates the files in ”My Drive” and selects a file for download in order to preview its content. |
| **Overview** | The user navigates the files in ”My Drive” and selects a file for download in order to preview its content. The system starts download it and decrypts the file if it was encrypted in the first place. The system also maintains the downloaded list for later use. |
| **Type** | Primary. |
| **Cross References** | Not Applicable. |

**Typical Course of Events**

| **Actor Action** | **System Response** |
|---|---|
| 1. This use case begins when the user clicks image/doc/pdf in the file list. | |
| | 2. The system starts downloading the file content and maintains a downloading task list. |
| | 3. The system gives a notification that download has completed. |

### 2.2.5 Use Case 5

| | |
|---|---|
| **Use Case** | Preview a Downloaded file. |
| **Actors** | User |
| **Purpose** | The user navigates the downloaded file list in ”Downloaded Task(s)” and verifies their face to view the file content. |
| **Overview** | The user navigates the downloaded file list in ”Downloaded Task(s)” and verifies their face to view the file content. |
| **Type** | Primary. |
| **Cross References** | Not Applicable. |

**Typical Course of Events**

| **Actor Action** | **System Response** |
|---|---|
| 1.This use case begins when the user clicks a file name from ”Downloaded Task(s). | |
| | 2. The system runs face verification prompt. |
| 3. The user aligns their face to the camera to verify the face. | |
| | 4. The system checks the face for verification against the reference picture and also runs anti-spoof mechanism. |
| | 5. Upon verification, the system gives the user a preview of the file. |

## 2.3 UI Designs (Prototype)

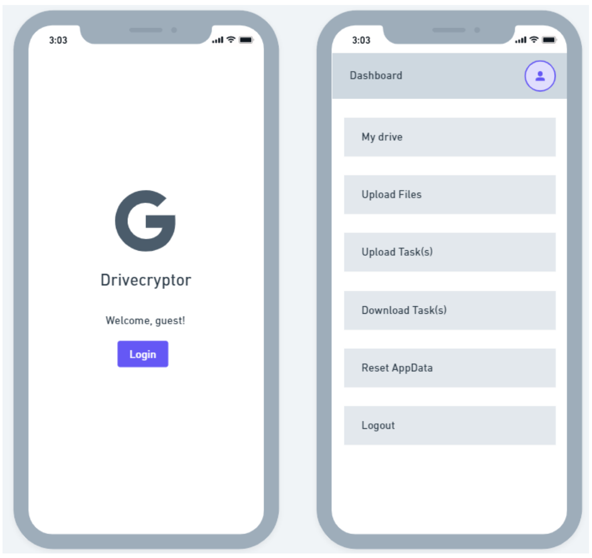


Figure 2: Login, Dashboard UI Prototype

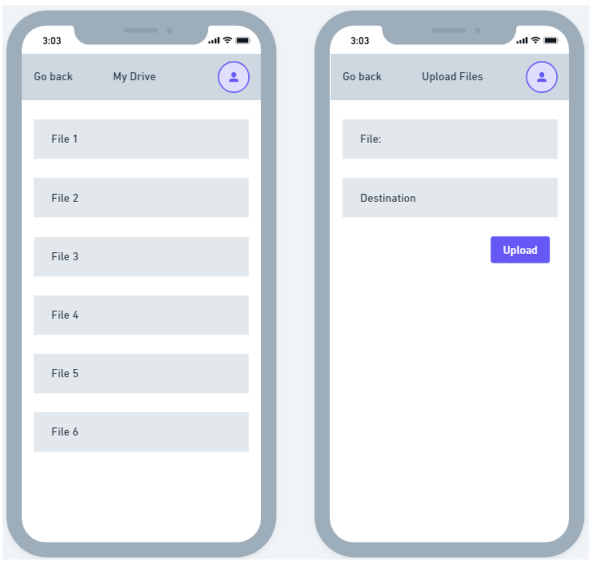


Figure 3: My Drive and Upload UI Prototype

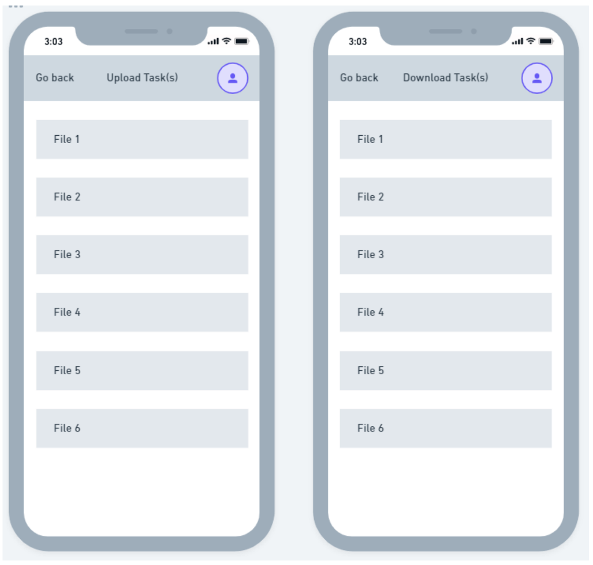


Figure 4: Uploaded & Downloaded Task(s) UI Prototype

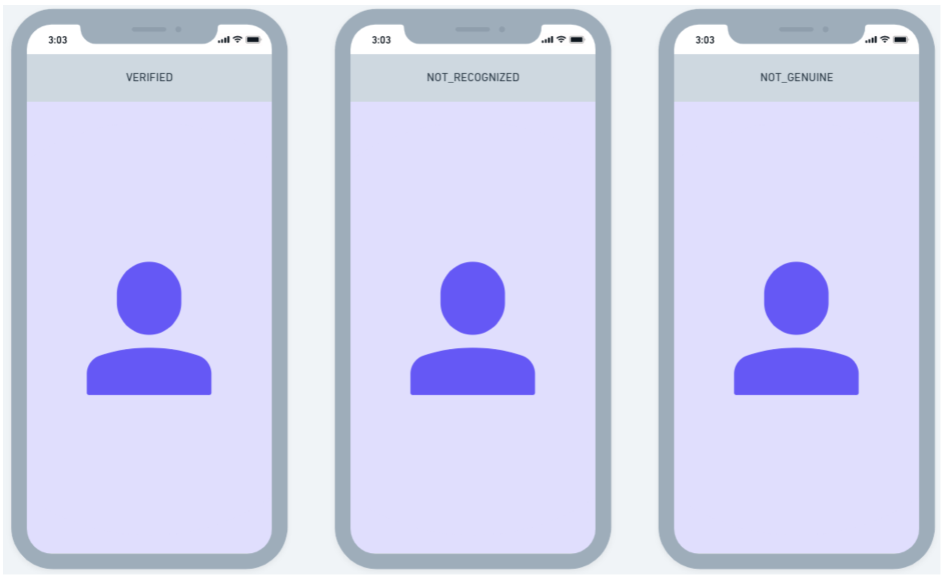


Figure 5: Face Verification UI Prototype

# 3. System Design

## 3.1 Database Design (ER Diagram)

The system is also designed by keeping in mind that storing and managing a database is unnecessary as **Google Drive's AppData space** securely provides the means to store our data: a list of all encrypted file names and reference pictures. **So, there is no traditional database in our app system, to begin with.**

## 3.2 API Documentation

For face recognition and anti-spoofing solution, we are using FaceOnLive. Also, we are utilizing a community-developed wrapper package to access Google Drive APIs. Finally, for an encryption-decryption solution, we chose cyptoJS's AES algorithm.

**FaceOnLive:** [https://docs.faceonlive.com/face-recognition/reference/android]

**GDrive API Wrapper:** [https://www.npmjs.com/package/@react-native-google-signin/google-signin]

**CryptoJS:** [https://github.com/brix/crypto-js/blob/develop/README.md]

## 3.3 Data Flow Diagrams

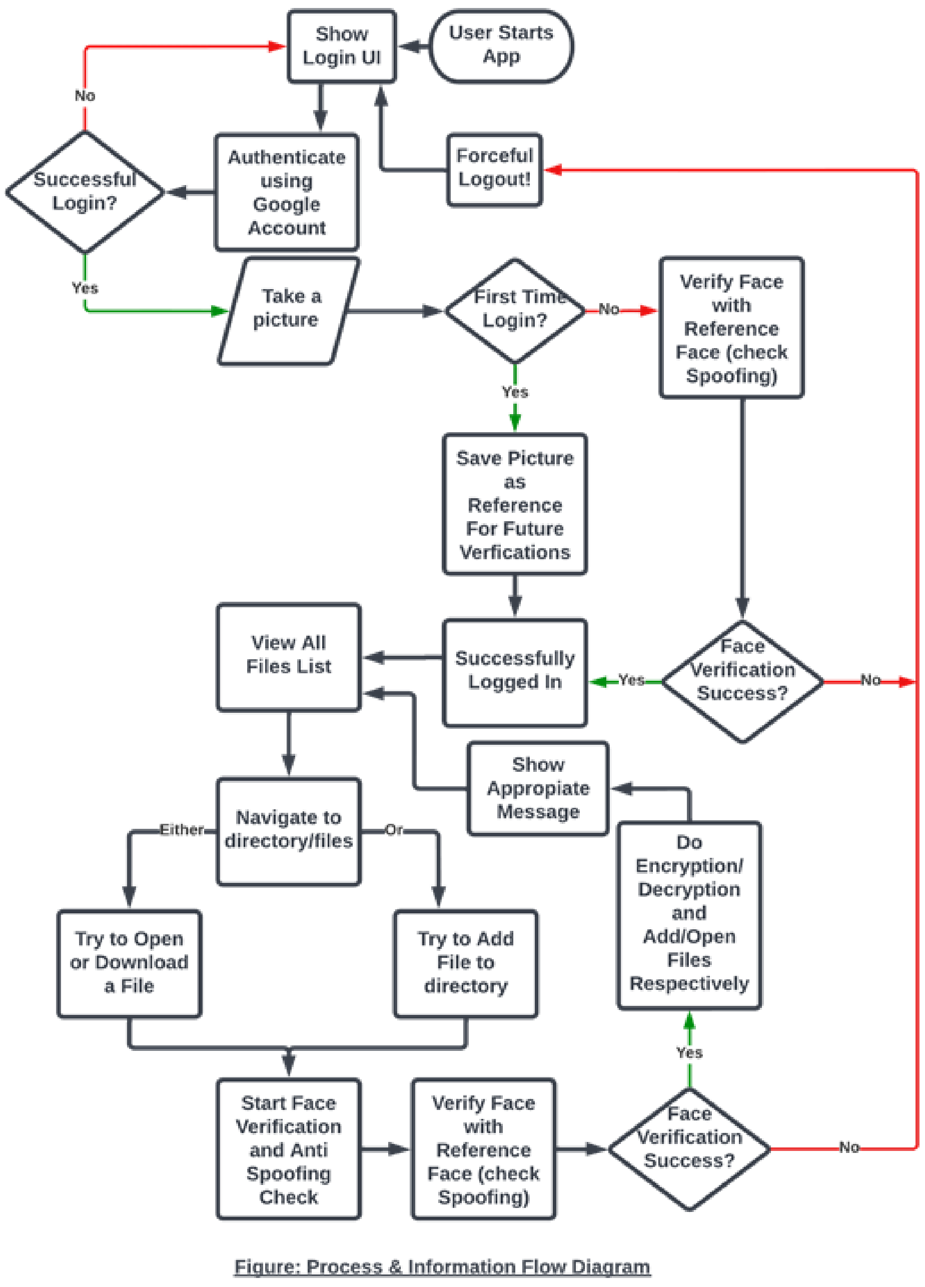

Show Login UI
User Starts App
No
Successful Login?
Authenticate using Google Account
Forceful Logout!
Yes
Take a picture
First Time Login?
No
Verify Face with Reference Face (check Spoofing)
Yes
Save Picture as Reference For Future Verfications
View All Files List
Successfully Logged In
Yes
Face Verification Success?
No
Show Appropriate Message
Navigate to directory/files
Either
Or
Do Encryption/ Decryption and Add/Open Files Respectively
Try to Open or Download a File
Try to Add File to directory
Yes
Start Face Verification and Anti Spoofing Check
Verify Face with Reference Face (check Spoofing)
Face Verification Success?
No
Figure: Process & Information Flow Diagram

Figure 6: Use Case Diagram

## 3.4 System Design Diagrams

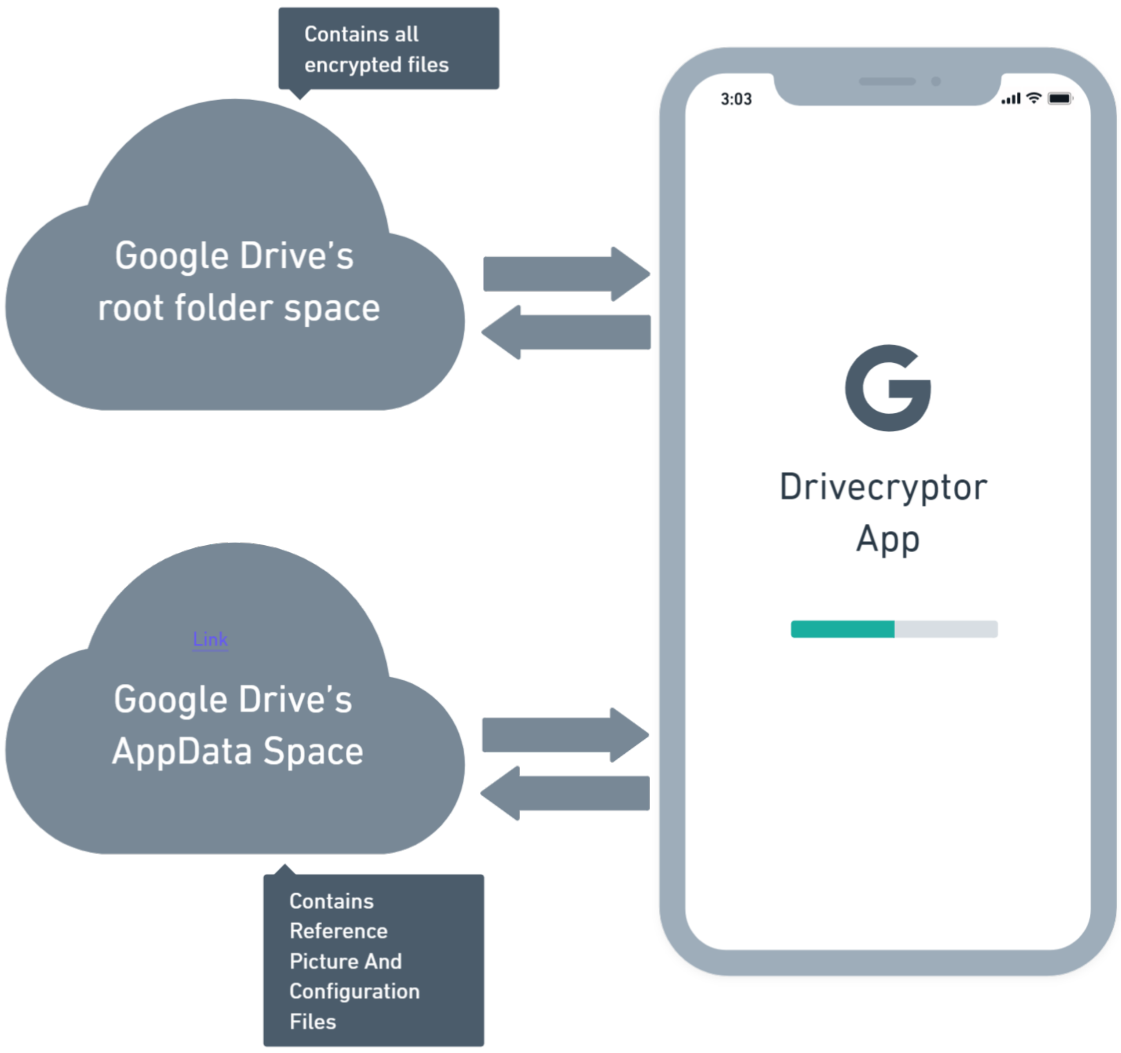

Contains all encrypted files
Google Drive's root folder space
3:03
Drivecryptor App
Link
Google Drive's AppData Space
Contains Reference Picture And Configuration Files

Figure 7: System Design Diagrams

# 4. Implementation Details

## 4.1 Work Breakdown

The work breakdown revolves around the use-cases implementation by our development team. The implementation deadline is of five weeks and five use cases. So, developers should implement each use case weekly to deliver before the deadline. However, upon considering the workload of the inner implementation details, the face recognition and anti-spoofing implementation plan are separated into two weeks. The uploading and downloading implementation are similar; hence, we are keeping them in the same week's workload.

## 4.2 Implementation Plan

- **Week 1:** Google sign-up/sign-in. Google Drive file list view.
- **Week 2:** Uploading and downloading file in Google Drive.
- **Week 3:** Taking photo and securely store it in Google Drive. Implementation of encryption and decryption.
- **Week 4:** Face recognition implementation.
- **Week 5:** Implementation of spoof detection.

## 4.3 Change Management

No further changes were made after the external evaluator's evaluation.

# 5. System Evaluation

## 5.1 Relevant Metrics

### 5.1.1 Face recognition

The face recognition component of the app uses an API that is based on a multi-class binary classification model. To evaluate how accurately this API recognizes faces, we use the following statistical metrics popularly used to measure the performance of such models: $F_1$ score and accuracy.

In our evaluation, the class is a person with one reference photo and multiple unique test photos. Each test is a determination for each class and for each test photo whether the face recognition API predicts that test photo to be in that class (positive) or not (negative). The result of a test falls into one of the following three categories:

- True positive (TP): Predicted positive, and the test photo belongs to the class
- False positive (FP): Predicted positive, but the test photo does not actually belong to the class
- False negative (FN): Predicted negative, but the test photo does actually belong to the class

The following is an explanation of the metrics we are using:

**$F_1$ score** The $F_1$ *score* is the harmonic mean of precision and recall.

- *Precision* is the fraction of relevant instances among the retrieved instances. It answers the question, "of all the positive predictions made, how many of them are truly positive?", and is calculated as the number of true positives (TP) divided by the total number of positives (i.e. true positives (TP) plus false positives (FP)):

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}.$$

- *Recall* is the fraction of relevant instances that were retrieved. It answers the question, "of all the actual positive examples out there, how many of them were predicted to be positive?", and is calculated as the number of true positives (TP) divided by the sum of the number of true positives (TP) and false negatives (FN):

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FN}}.$$

For a comprehensive evaluation of the classifier's performance, both precision and recall should be examined simultaneously, especially if the dataset is imbalanced. By incorporating both, the $F_1$ score provides a more balanced and, hence, more helpful summarization of model performance than either of them alone.

For our evaluation, we use both the macro- and the micro-averages of these metrics (precision, recall, $F_1$ score). The macro-averages are computed by first computing the metrics for each class and then averaging them. The micro-averages are computed using the total numbers of TP, FP, and FN for the entire dataset, as if computing the macro-averages for a dataset with only one class.

**Accuracy** *Accuracy* measures how well the binary classification model correctly identifies or classifies a test object. It is calculated as the proportion of correct predictions (both true positives and true negatives) among the total number of cases examined:

$$\text{accuracy} = \frac{\text{TP}}{\text{total number of test cases}}.$$

In a binary classification test where the classifier associates each test object with a single class, as it is in our test, the accuracy has the same value as the micro-averaged $F_1$ score.

## 5.2 Evaluation Methods

### 5.2.1 Face recognition

For evaluating our face recognition system, we collected a set of photos containing the faces of 20 individuals at roughly four photos for each. For each person, we selected photos with different backgrounds, lighting conditions, and angles. We developed an app that runs our face recognition system on this dataset and computes our required statistical metrics.

### 5.2.2 Face spoof detection

We collected 5 different inputs/cases that caused our spoof detection to give wrong results. Each of these inputs was either a photo or a video of a person's face captured using a modern high-resolution camera, which was fed as input to the spoof detection app running on an android phone. Let's call this phone the detecting phone. The input was fed through the detecting phone's camera once the app was started. The following conditions were common for all the failing cases:

- The person's face was at an approximate distance of 0.5 meters from the camera.
- The person was looking straight at the camera with no tilting or leaning.
- The person's face had either a neutral emotion or a slight smile.
- The person was looking at the camera as if to unlock a face lock.
- The picture was taken under well-lit conditions.
- The detecting phone's camera was not catching light reflecting off the screen that was displaying the picture.

The cases are described further below.

| | **Input** | **Expected output** | **Actual output** |
|---|---|---|---|
| **Case #1** | A still picture of person 1's face, with the detecting phone capturing the input (the picture displayed on another device) from an approximately 0.25 meter distance at a slightly tilted angle | "Fake" (spoof) | "Real" (live) |
| **Case #2** | A 3-second video of person 2's face | "Fake" (spoof) | "Real" (live) |
| **Case #3** | A 3-second video of person 2's face partly covering his mouth with his hand | "Fake" (spoof) | "Real" (live) |
| **Case #4** | A 3-second video of person 3's face wearing a mask | "Fake" (spoof) | "Real" (live) |
| **Case #5** | A 3-second video of person 4's face wearing a mask | "Fake" (spoof) | "Real" (live) |

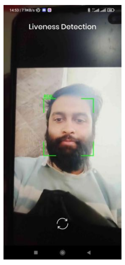


Figure 8: Case 1 screenshot

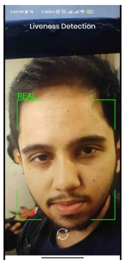


Figure 9: Case 2 screenshot

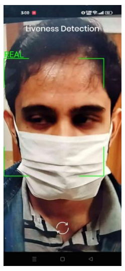


Figure 10: Case 3 screenshot

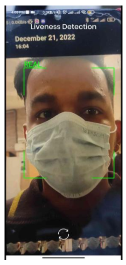


Figure 11: Case 4 screenshot

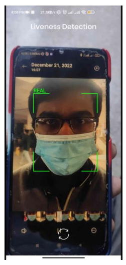


Figure 12: Case 5 screenshot

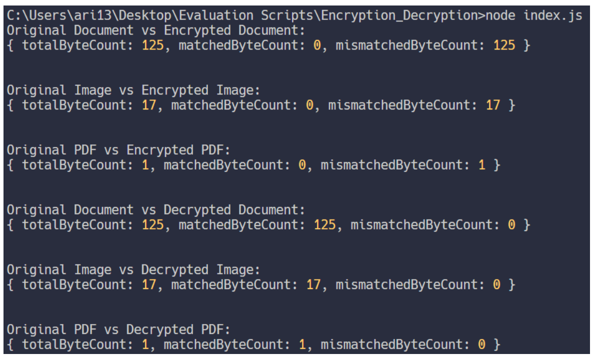
### 5.2.3 Usability evaluation

For this task, we collaborated with another group. The other group conducted the actual usability check. For each of the use cases we have listed earlier, the testing team was asked to perform the use case and keep count of the number of inputs. The number of interactions required to carry each use case was noted. For accuracy, the test was repeated three times for each use case. The results were tabulated. The evaluator's document is attached in the appendix of this report.

## 5.3 Evaluation Implementation Details

### 5.3.1 Encryption and Decryption

We created an evaluation script where we verified files of different types, i.e. documents, images and PDF to test our CryptoJS's AES Algorithm for encryption and decryption. Two sets of files were prepared, one is the original file and the other is the encrypted file. The test was performed through bit by bit comparison between the original file and the encrypted file. Figure 8 below shows the result of the evaluation.

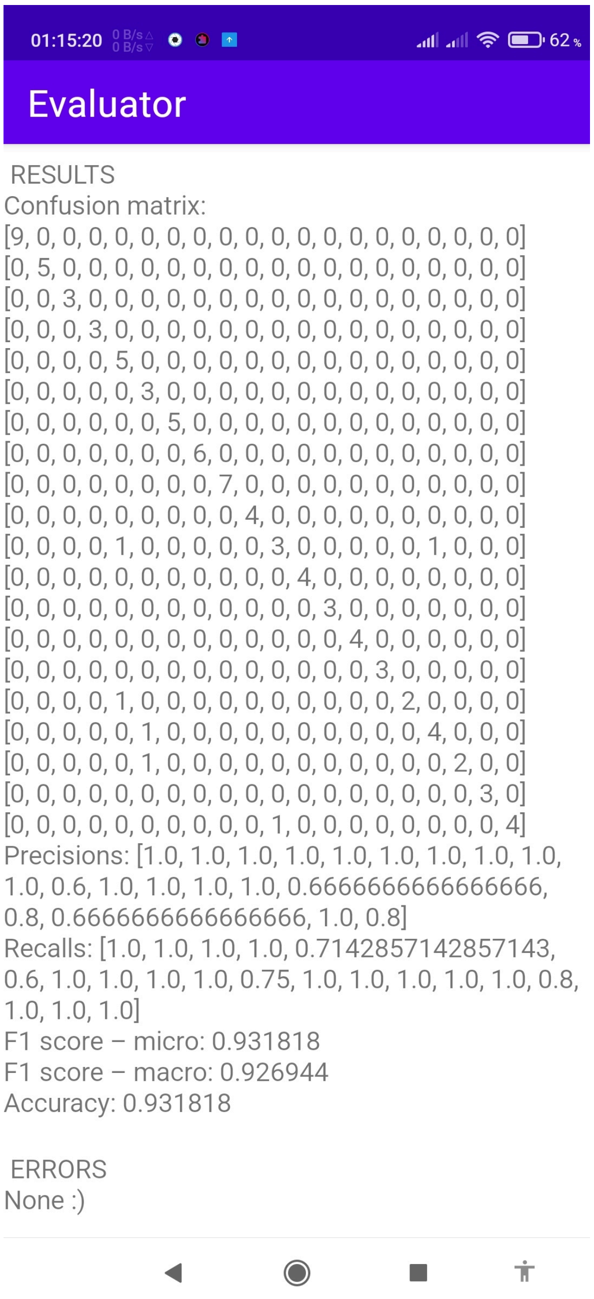


```
C:\Users\ari13\Desktop\Evaluation Scripts\Encryption_Decryption>node index.js
Original Document vs Encrypted Document:
{ totalByteCount: 125, matchedByteCount: 0, mismatchedByteCount: 125 }


Original Image vs Encrypted Image:
{ totalByteCount: 17, matchedByteCount: 0, mismatchedByteCount: 17 }


Original PDF vs Encrypted PDF:
{ totalByteCount: 1, matchedByteCount: 0, mismatchedByteCount: 1 }


Original Document vs Decrypted Document:
{ totalByteCount: 125, matchedByteCount: 125, mismatchedByteCount: 0 }


Original Image vs Decrypted Image:
{ totalByteCount: 17, matchedByteCount: 17, mismatchedByteCount: 0 }


Original PDF vs Decrypted PDF:
{ totalByteCount: 1, matchedByteCount: 1, mismatchedByteCount: 0 }
```

Figure 13: Results of evaluation of encryption and decryption

### 5.3.2 Face recognition

One can follow our process for evaluating the face recognition system by running the app Evaluator included with the project files. Following the on-screen instructions, the user places the test photos in a specific directory on the Android device (a set of photos for 20 individuals, which we used for our evaluation, is provided with the project files). The app then runs the face recognition system on those photos and generates a confusion matrix (normalizing the dataset by individuals). From that confusion matrix, it computes the following statistical metrics: precision, recall, and $F_1$ score – both the micro-average and the macro-average of each – and the accuracy. The app displays the results and, if the user selects the 'Save Results to File' button, saves the result in two files – the confusion matrix in `confmat.txt` and the metrics in `metrics.txt` – both located in the same directory as the parent of the directory containing the photos.

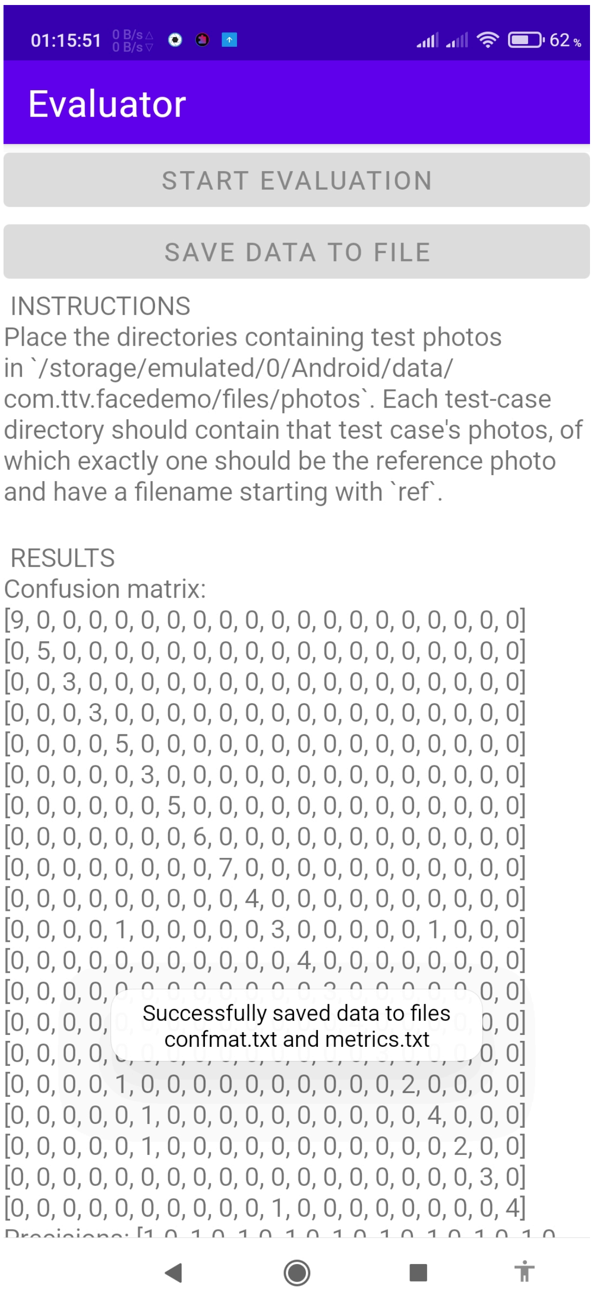


```
RESULTS
Confusion matrix:
[9, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 1, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4, 0, 0, 0]
[0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 4]
Precisions: [1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
1.0, 0.6, 1.0, 1.0, 1.0, 1.0, 0.6666666666666666,
0.8, 0.6666666666666666, 1.0, 0.8]
Recalls: [1.0, 1.0, 1.0, 1.0, 0.7142857142857143,
0.6, 1.0, 1.0, 1.0, 1.0, 0.75, 1.0, 1.0, 1.0, 1.0, 1.0, 0.8,
1.0, 1.0, 1.0]
F1 score – micro: 0.931818
F1 score – macro: 0.926944
Accuracy: 0.931818

ERRORS
None :)
```

Figure 14: Result of face recognition evaluation

## 5.4 Summary of Evaluation

The verification of the system was carried out to assure the correctness of the system in terms of requirements and technical errors. The accuracy of the face recognition API was tested on our dataset containing images of 20 different individuals. We collected a set of inputs for our spoof detection API. The authenticity of the encryption and decryption algorithm was inspected by comparing the original file with the encrypted file. Moreover, validation was also conducted to get feedback from the potential users of the system. Concluding, the successful verification and validation of the system reflects that the requirements proposed for the software were taken into account and were well integrated into the system.



INSTRUCTIONS
Place the directories containing test photos in `/storage/emulated/0/Android/data/com.ttv.facedemo/files/photos`. Each test-case directory should contain that test case's photos, of which exactly one should be the reference photo and have a filename starting with `ref`.

RESULTS
Confusion matrix:
```
[9, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 1, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0]
[0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4, 0, 0, 0]
[0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 4]
Precisions: [1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0
```

Successfully saved data to files confmat.txt and metrics.txt

Figure 15: Result of face recognition evaluation

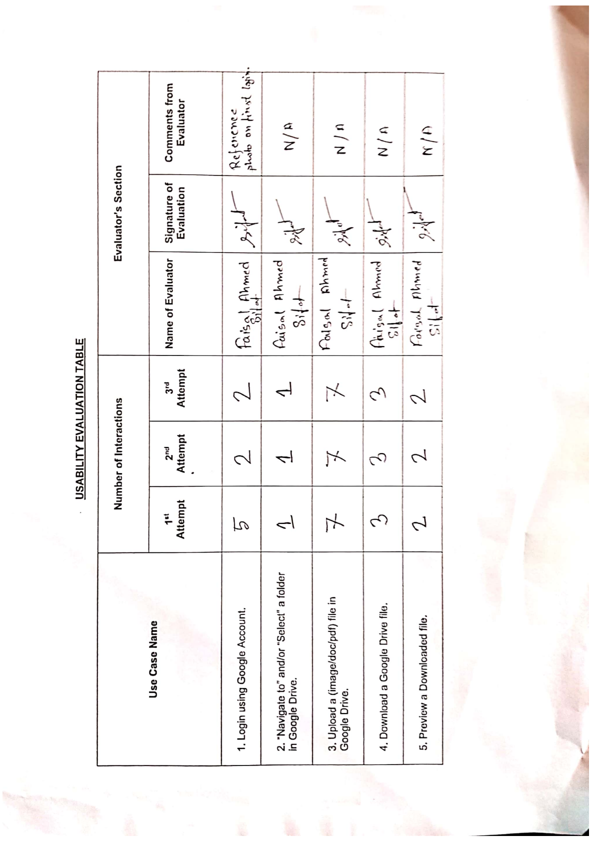
# 6. Conclusion

The main purpose of this paper was to provide a vivid elucidation of the **Drivecryptor App** to the intended audience. The introductory part provided the reader with a glimpse of the application mentioning its core features. The system feature part unfolded the use case of the system with the help of a use case diagram and in further details using an expanded use case. Furthermore, the UI design showcased the simplicity and consistency maintained to minimize actions per screen. In addition, the system design part included the API documentation, Data flow diagram and System Design diagram of the application. The next section presented the implementation details through work breakdown methodology and implementation plan. The last section depicted the system evaluation that was conducted to check for the correctness and usability of the application. To conclude, the report provided the reader with an amalgamation of software requirements specification (SRS) and software design specification (SDS) along with system evaluation of the software.

# A. Appendix

## A.1 External Evaluator Generated Documents

See the attached document in the next page.



**USABILITY EVALUATION TABLE**

| Use Case Name | Number of Interactions | | | Evaluator's Section | | |
|---|---|---|---|---|---|---|
| | 1st Attempt | 2nd Attempt | 3rd Attempt | Name of Evaluator | Signature of Evaluation | Comments from Evaluator |
| 1. Login using Google Account. | 5 | 2 | 2 | Faisal Ahmed Sifat | [illegible] | Reference photo on first login. |
| 2. "Navigate to" and/or "Select" a folder in Google Drive. | 1 | 1 | 1 | Faisal Ahmed Sifat | [illegible] | N/A |
| 3. Upload a (image/doc/pdf) file in Google Drive. | 7 | 7 | 7 | Faisal Ahmed Sifat | [illegible] | N/A |
| 4. Download a Google Drive file. | 3 | 3 | 3 | Faisal Ahmed Sifat | [illegible] | N/A |
| 5. Preview a Downloaded file. | 2 | 2 | 2 | Faisal Ahmed Sifat | [illegible] | N/A |

Figure 16: Results of evaluation of encryption and decryption