



**VIT**<sup>®</sup>  
**Vellore Institute of Technology**  
(Deemed to be University under section 3 of UGC Act, 1956)

## **CSE3502 : Information Security Management**

### **Project Review 2**

**Data Protection and encryption security in cloud**

Slot : F2

#### **Team Members :**

<b>Name</b>	<b>Registration Number</b>
Deepak Agarwal	18BIT0437
Kanishak Garg	18BIT0434

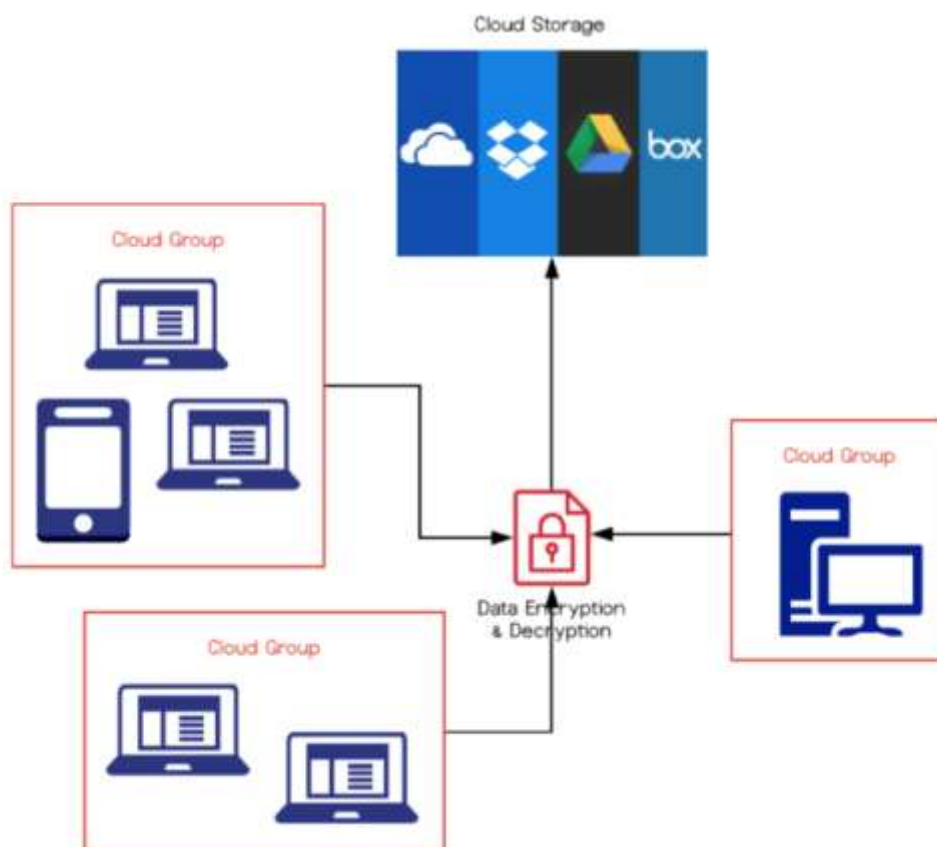
## **Abstract :**

The aim of this project is to develop a secure cloud storage application for Dropbox Box, Google Drive etc. The application is able to secure all files that are uploaded to the cloud, such that only people that are part of your “Secure Cloud Storage Group” will be able to decrypt your uploaded files. To all other users the files will be encrypted. A suitable key management system will be designed and implemented for the application that will allow files to be shared securely, and users to be added and removed from “Secure Group”. The application make use of open source cryptographic libraries. The implementation of this assignment is divided into mainly two components: User Interface and a Cloud Storage Group. These components make use of different modules that consists of helper functions to execute their tasks. The highlight of the design revolves around how to transmit symmetric keys to valid users in the group securely. The symmetric keys used for encryption when uploading files and decryption when downloading. The contents in the cloud storage has to be encrypted for data protection and can only be decrypted by users who have access to the symmetric keys. To achieve this, a symmetric key has to shared securely among the users. The initiative taken here is to have the users send their public keys to the cloud storage group, so that it can be used to encrypt the symmetric key during transmission.

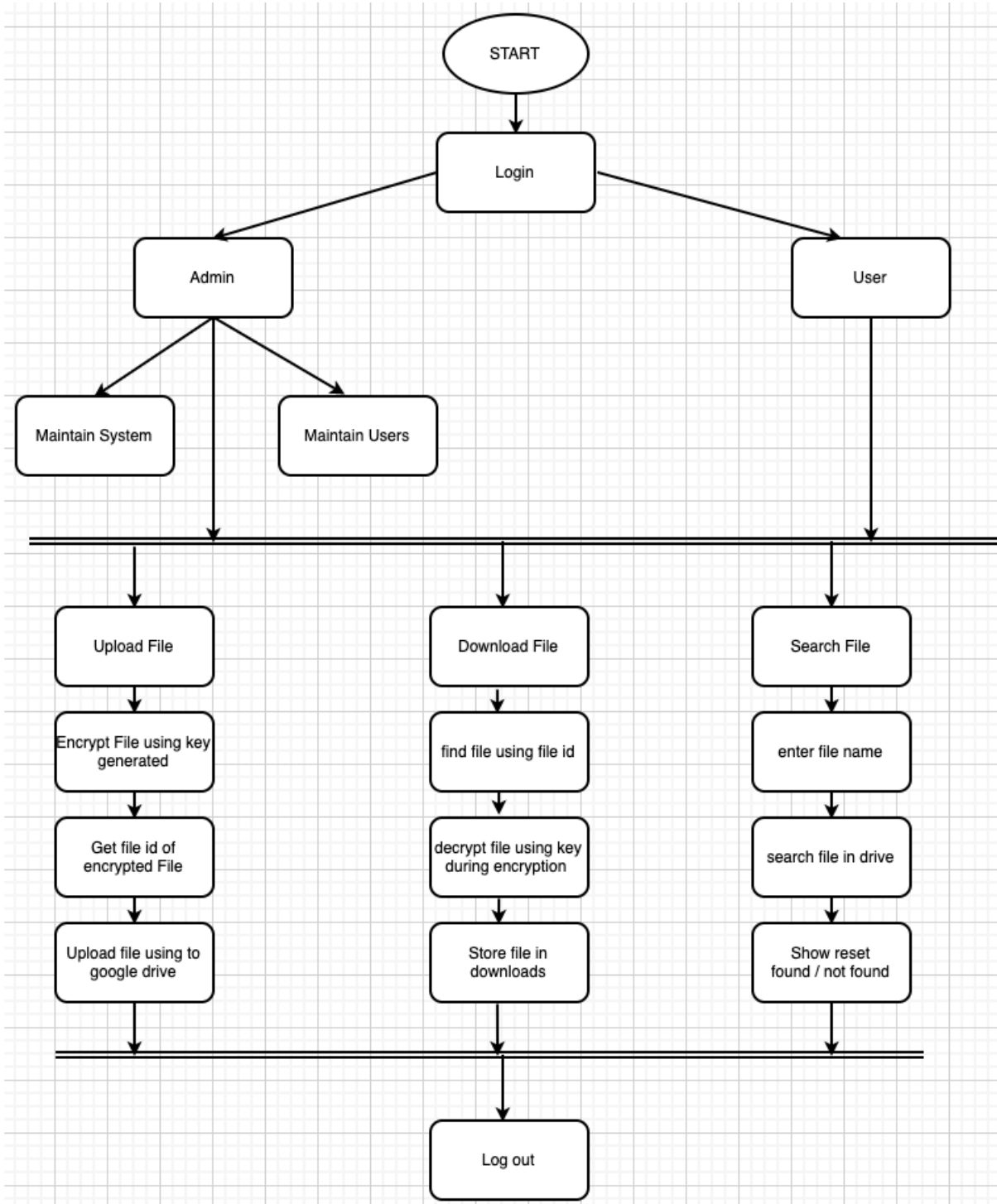
## Design:

For implementation of this project, we divided it into mainly two components: User Interface and a Cloud Storage Group. These components make use of different modules that consist of helper functions to execute their tasks. The highlight of the design revolves around how to transmit symmetric keys to valid users in the group securely. The symmetric keys used for encryption when uploading files and decryption when downloading. One critical assumption that is taken on during the development of the application is such that the activities of the cloud storage group is always being peeked on and the contents in it is always exposed to outsiders.

## High level design :



# Low Level Design:



# Modules Implemented :

## 1. User Interface:

### a. Main.py File:

Main() : When the program starts the very first thing it does is look to see if a key exists, if no key exists then one is generated automatically. Next it checks to see if there are any previous users registered in the program. If there are no users, then it prompts the user to create an admin account which will be given privileged rights to create new accounts or generate new keys. Otherwise it would present the user to login. If you log in as admin then you will have a menu specific to that admin user with added functionality. Otherwise you get presented with a standard menu. This is running in a loop to allow users to log in and out as they wish.

### b. Users.py

Save\_list(obj, name) :

This function is used to pickle (save) the list of users that are part of the program. I use a dictionary to store the user name and password as a key value pair. It is called every time a new user is created.

load\_list(name) :

This function is used to load the object where the usernames and passwords are stored. It is called when the user class is invoked to ensure the usernames are up to date.

newUser() :

This function is used to create a new user. It is simple as it asks the user to enter a user name, it warns the user if this user name exists and then asks for a password and stores it in our dictionary and pickles it.

Login() :

This function is called in the main line, it provides the user with a login screen and checks if the user logging in is the admin, a standard user or if they fail to login in 3 attempts it will terminate the session.

deleteUser() :

This is called to delete a user by the admin. It prevents from the admin account from being deleted and also if the user does not exist it will notify the user. Once a certain user is deleted the username dictionary is once again pickled and stored safely

## 2.Cloud storage group

### a. GoogleDriveAPI.py

Here there are all relevant functions required to carry out uploading, downloading and file searching. The functions include:

- uploadFile(fileName)
- downloadFile(file\_id, fileName)
- searchFile(query)
- fileID(query)

### b. MenuTypes.py

PrivilegeMenu() :

This generates a menu for the admin which has more features such as adding and deleting users and also generating a new key. This is where the majority of actions are performed such as uploading a file, it will encrypt then upload and when downloading from the drive it will download and decrypt the file.

StandardMenu():

This menu is nearly identical to the above menu however the settings option is removed to prevent the user from performing privileged actions.

**c. Auth.py:**

This is used to authenticate with Google Drive. This is taken from the Google Drive API QuickStart guide.

**d. Encryption.py**

KeyGen()

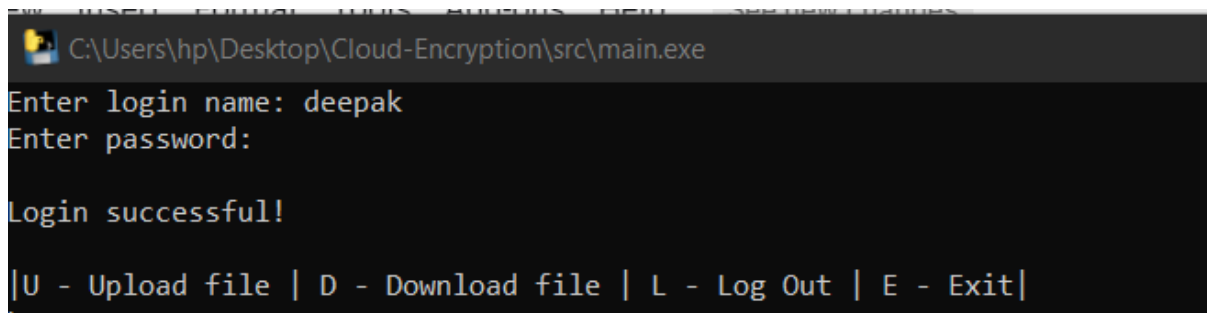
KeyGen as the name suggests is used to generate a new key and store it on the computer. KeyRead() This is used to read the key from the computer, if no key exists it will automatically generate a new one by calling keyGen().

Encrypt(filename, Key) This is used to encrypt the file, it is using Fernet to perform the required encryption.

Decrypt(filename, key) This is used to decrypt the file that had been encrypted.

## Screenshots of the system:

**Login functionality** :- first user needs to log in before he is authorized to access the files or any functionality.

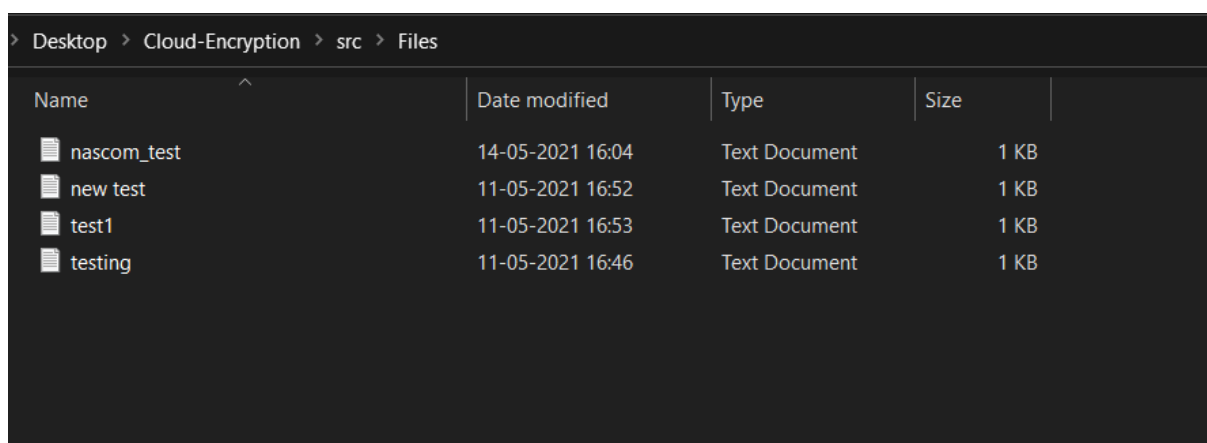






```
C:\Users\hp\Desktop\Cloud-Encryption\src\main.exe
Enter login name: deepak
Enter password:

Login successful!

|U - Upload file | D - Download file | L - Log Out | E - Exit|
```

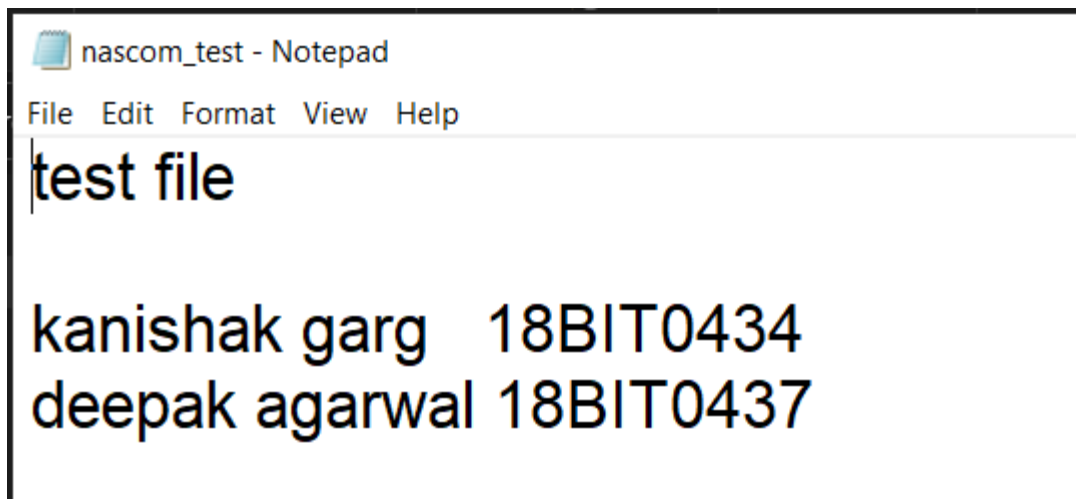
**Uploading a file :-** for uploading a file, file need to be placed in the files folder from where we will take the file to upload to the google drive.



> Desktop > Cloud-Encryption > src > Files				
Name	Date modified	Type	Size	
 nascom_test	14-05-2021 16:04	Text Document	1 KB	
 new test	11-05-2021 16:52	Text Document	1 KB	
 test1	11-05-2021 16:53	Text Document	1 KB	
 testing	11-05-2021 16:46	Text Document	1 KB	



Text written in file before uploading:



Upload the file to the drive and a file id is generated. File will be in an encrypted format in the google drive and will be decrypted at time of decryption:

```
|U - Upload file | D - Download file | L - Log Out | E - Exit|
U

Ensure that files you wish to upload are in the 'Files' folder.
Enter m to return to main menu.

Enter file name: nascom_test.txt
new_request() takes at most 1 positional argument (2 given)
File ID: 10ZvRna0RFYvJnnX8L6sfwEy_IDqRHL6T

Ensure that files you wish to upload are in the 'Files' folder.
Enter m to return to main menu.

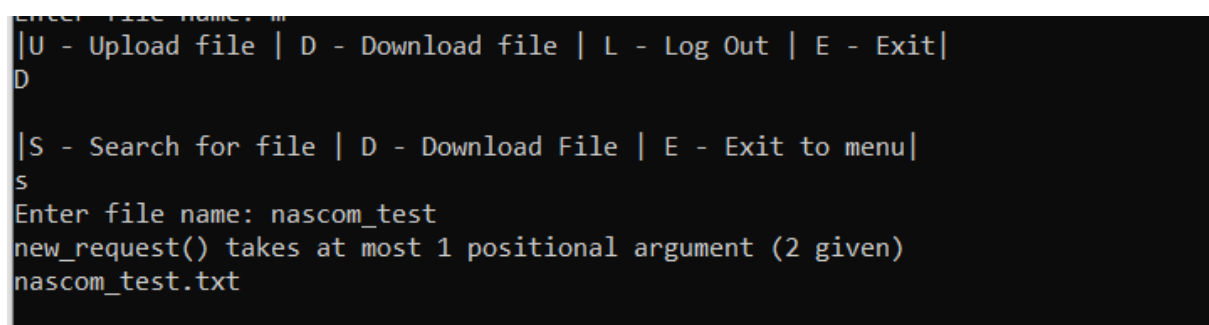
Enter file name:
```

File after getting uploaded in the google drive. It got encrypted. If anyone try to view it he/she will see only encrypted file :



```
nascom_test - Notepad
File Edit Format View Help
gAAAAABgnlljekrVVIUZH3-gjDD0MVmZI8Rmy-
xcWyfMJDcBm5pshkWcEVWSt5rk4FwSSs7xgMIppiMnpbAael-
BxTwVt9rvJ0EsetU0NofCSjgtTpGOqaQZL3_XYUdQfTMn-
G3d4W82FH0eCw_Xh3WCoR19YXf2rnX1lmQZliqAaTfwQ9hOXds=
```

Searching for the file in the drive:



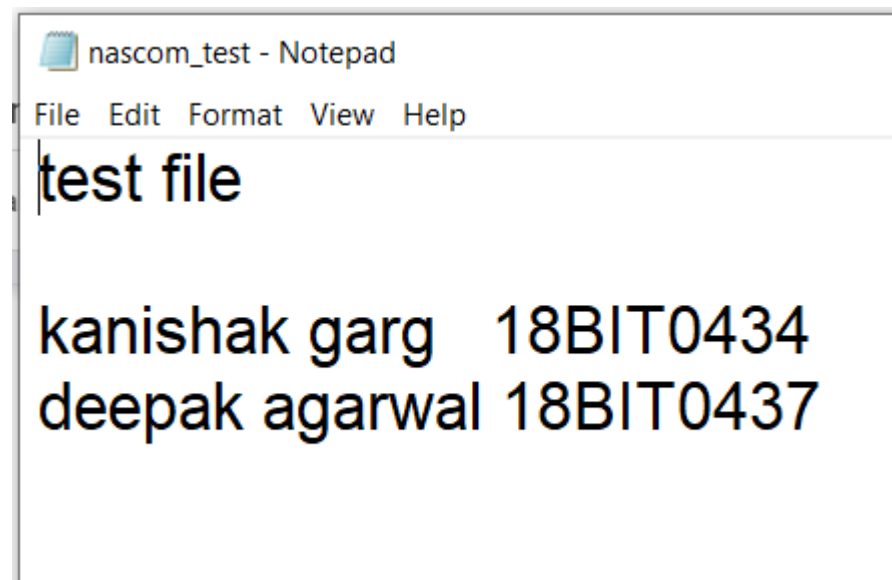
```
Enter file name: in
|U - Upload file | D - Download file | L - Log Out | E - Exit|
D

|S - Search for file | D - Download File | E - Exit to menu|
S
Enter file name: nascom_test
new_request() takes at most 1 positional argument (2 given)
nascom_test.txt
```

**Downloading the file** from the drive it will get downloaded in the downloads folder of the project and it will be available in an normal format means it will be decrypted.

```
|S - Search for file | D - Download File | E - Exit to menu|
d
Enter file name: nascom_test
new_request() takes at most 1 positional argument (2 given)
new_request() takes at most 1 positional argument (2 given)
Download 100%.
```

File after getting downloaded



**Log out functionality**

```
|U - Upload file | D - Download file | L - Log Out | E - Exit|
l
Logging out.
Enter login name:
```

## **Work for review 3:**

Once we had completed the working console application, then we decided to modify the functions above a little bit to allow for a seamless graphical user interface integration.

This means that we will create a web app where the UI is based on HTML, CSS & Javascript and the backend is still running on python.

We will try to make a web application so that Users can interact with the website in a better way.

We will also add the users functionality in which admin of the group can add or delete users from the secure cloud storage group.

He will have access to all the controls implemented over the system and can control who can see, edit or delete files.