**EX NO: Storage as a Service Configuration using DropBox**

**Date:**

**AIM:**

To implement the Storage as a Service we configuration processes in Dropbox in which we can create folders and upload our files.

**PROCEDURE:**

1)Create an account and Create new application in your account

2) Specify the type of access in the next window and Generate the ACCESS TOKEN for that particular application it will use for future programming phase:

3) Change the permission settings

4) Also note down the Application key and Application Secret Key as well:

5) Give “No Expiration” in Access token expiration and Generate a token access

6) Uploading the files in dropbox using python

**CODE:**

**Code:**

**import sys**

**import dropbox**

**from dropbox.files import WriteMode**

**from dropbox.exceptions import ApiError, AuthError**

**import datetime**

**dt = datetime.datetime.today()**

**TOKEN = 'sl.BH5LkK03aRLRLVw1PFEZjRNSbgKEjmH2D\_iJlAiDHDYFWQT0wYgFy1zjGBianKplXs1hWbBszVgkUoMgv4rLQ8qrfc4Ifg-TsSHvDjRLaIi3179z76vCBW1q8MA\_3ydp9f2UWnM'**

**LOCALFILE = 'C:\\Users\\rajna\\Downloads\\GOOGLECLOUDPROGRAM\\kitten.png'**

**# Don't forget to add the file extension at the end of BACKUPPATH.**

**BACKUPPATH ='/kitten.png'**

***def* backup(*localFile*, *backupPath*):**

**with open(*localFile*, 'rb') as f:**

**# We use WriteMode=overwrite to make sure that the settings in the file**

**# are changed on upload**

**print("Uploading " + *localFile* + " to Dropbox as " + *backupPath* + "...")**

**try:**

**dbx.files\_upload(f.read(), *backupPath*, *mode*=WriteMode('overwrite'))**

**except ApiError as err:**

**# This checks for the specific error where a user doesn't have**

**# enough Dropbox space quota to upload this file**

**if (err.error.is\_path() and**

**err.error.get\_path().reason.is\_insufficient\_space()):**

**sys.exit("ERROR: Cannot back up; insufficient space.")**

**elif err.user\_message\_text:**

**print(err.user\_message\_text)**

**sys.exit()**

**else:**

**print(err)**

**sys.exit()**

**if \_\_name\_\_ == '\_\_main\_\_':**

**# Create an instance of a Dropbox class, which can make requests to the API.**

**print("Creating a Dropbox object...")**

**dbx = dropbox.Dropbox(TOKEN)**

**# Check that the access token is valid**

**try:**

**dbx.users\_get\_current\_account()**

**except AuthError:**

**sys.exit("ERROR: Invalid access token; try re-generating an "**

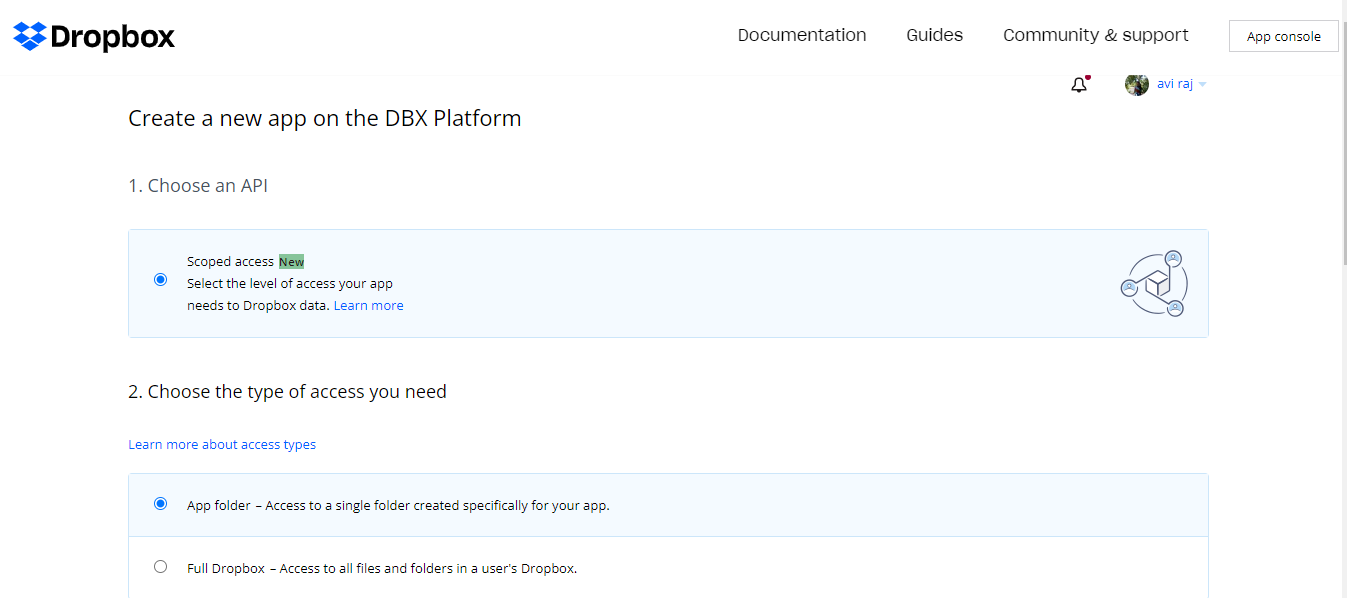
**"access token from the app console on the web.")**

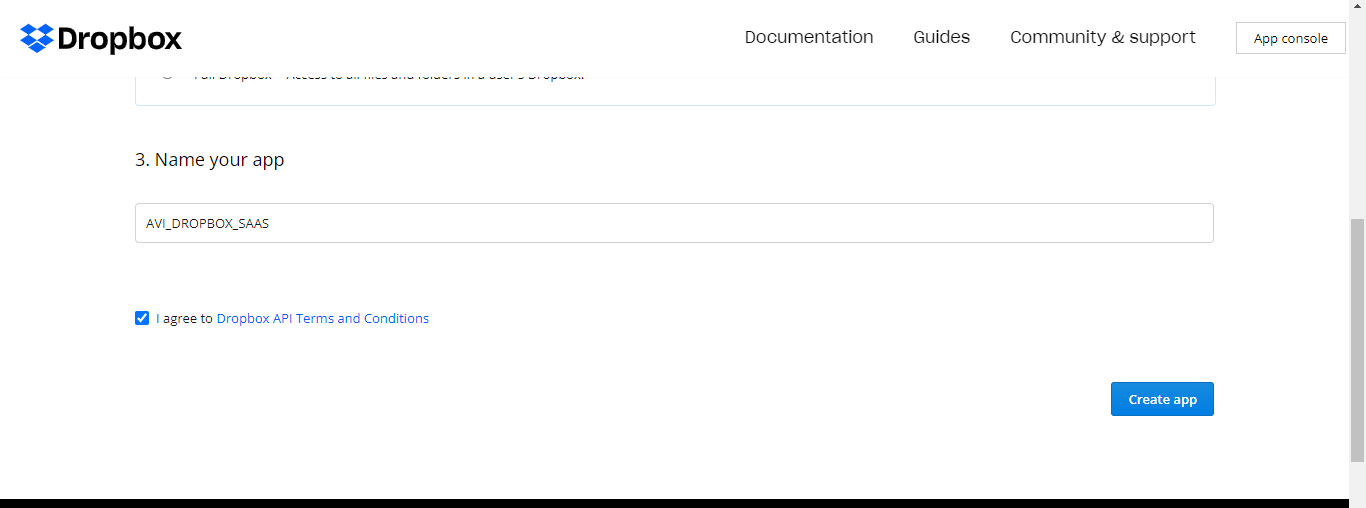
**# Create a backup of the current settings file**

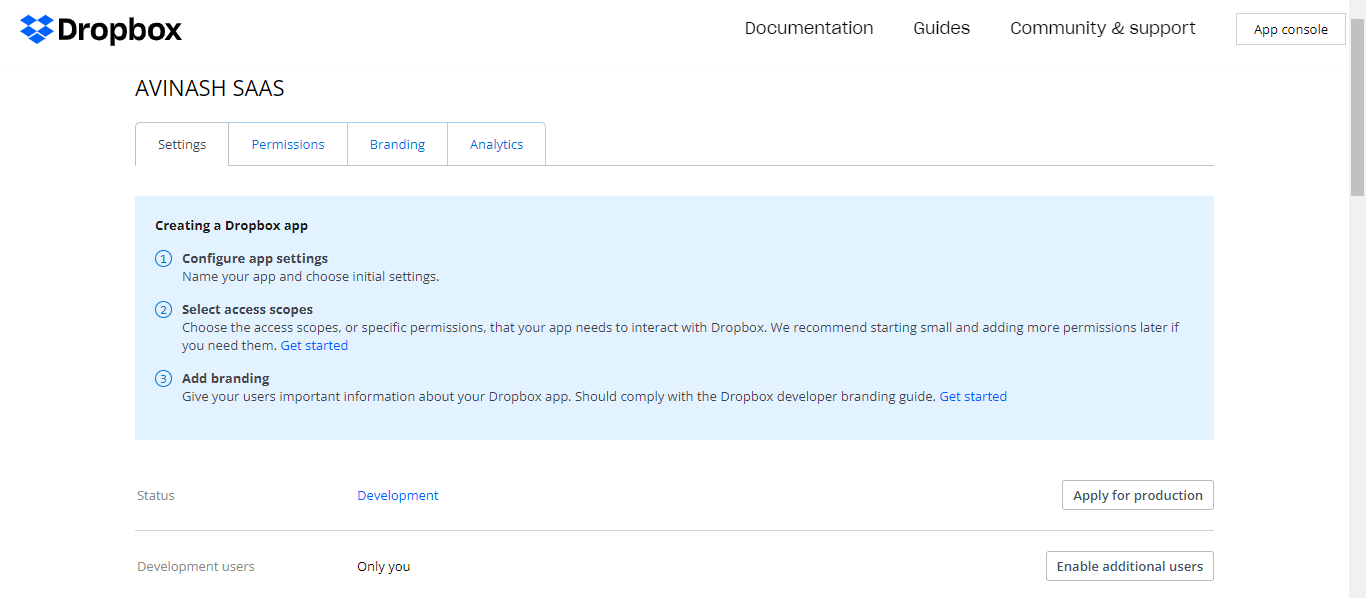
**backup(LOCALFILE, BACKUPPATH)**

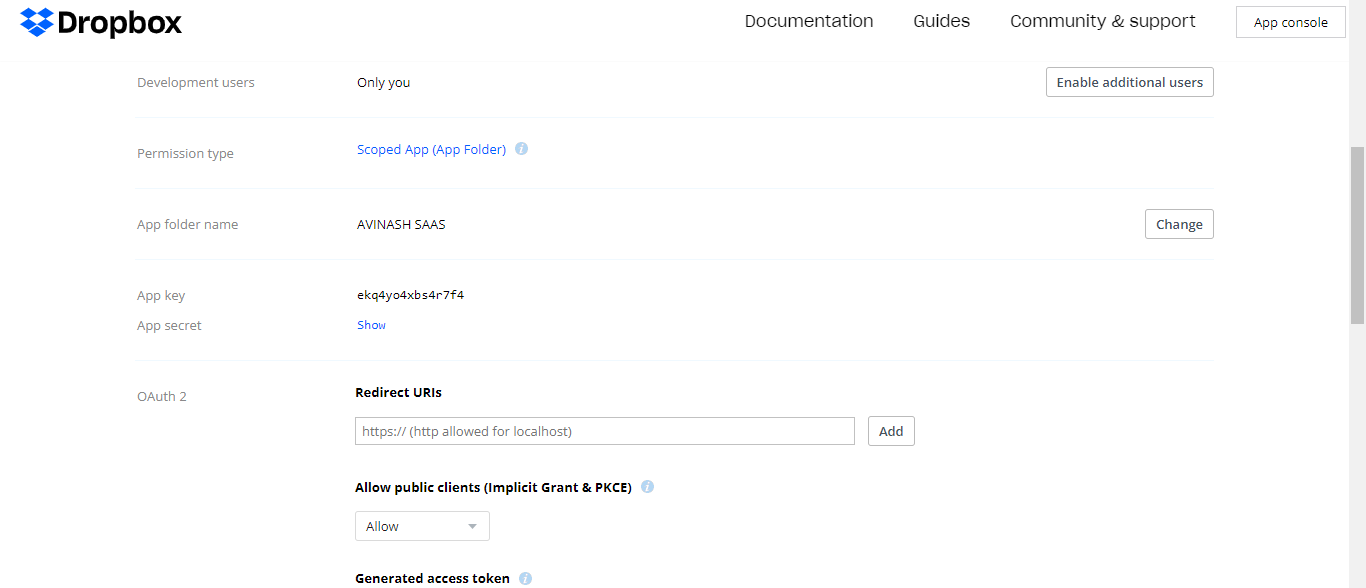
**print("Done!")**

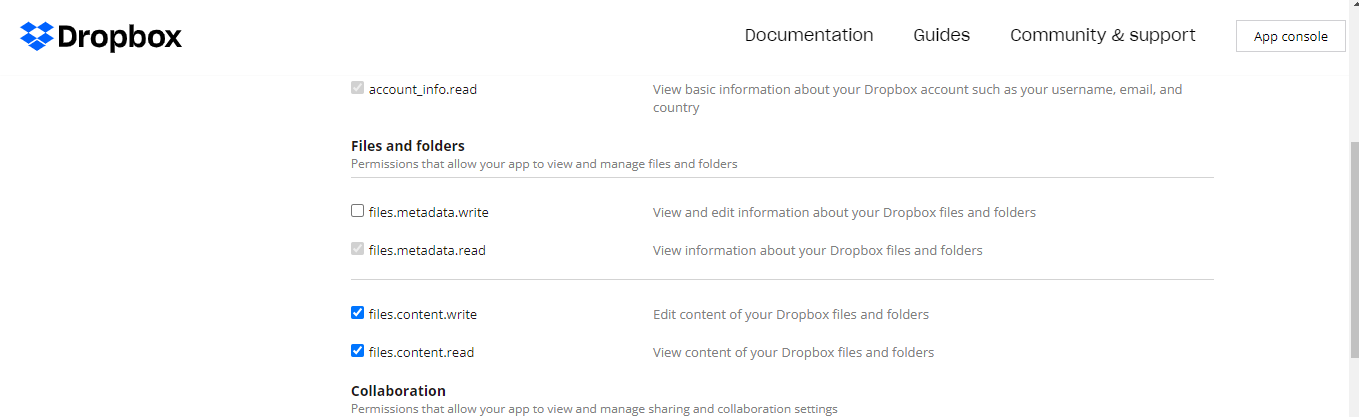
**SCREENSHOTS:**

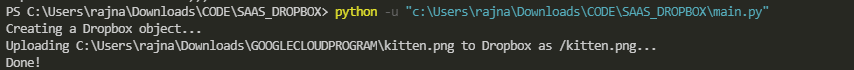
****

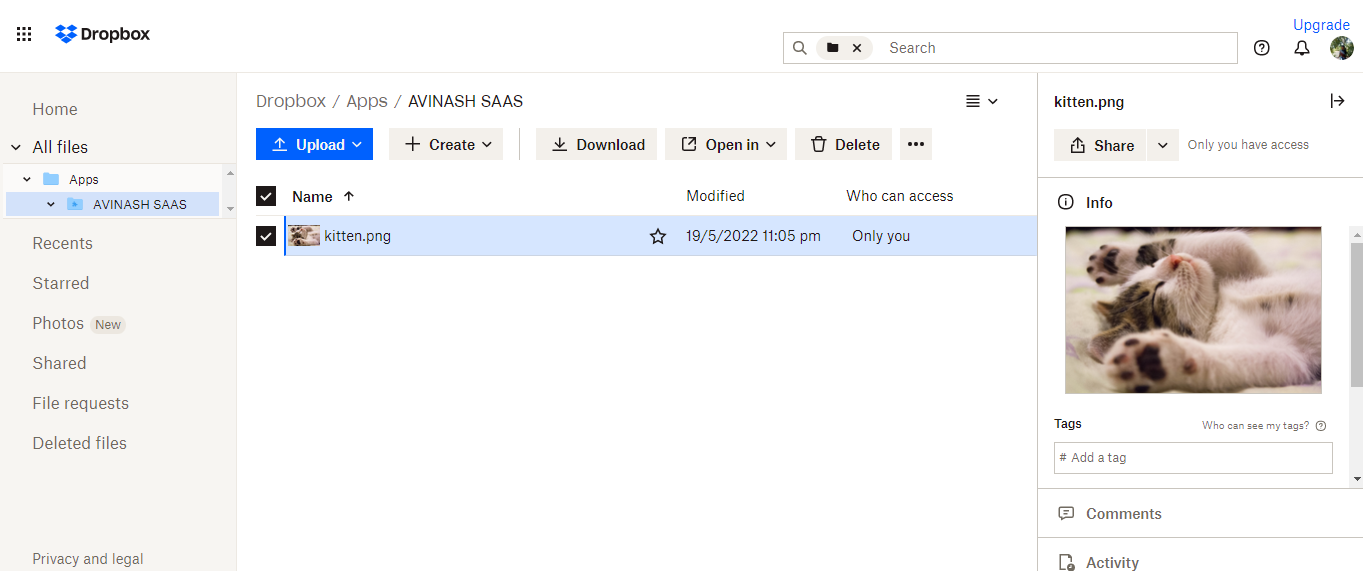
****

****

****

****

****

****

**Result:**

Thus, the Storage as Service using Dropbox is implemented by creating the python project to access the dropbox console and view all the files and folders which are created in it.