Python Code

```python
import os
from pyftpdlib.authorizers import DummyAuthorizer
from pyftpdlib.handlers import FTPHandler
from pyftpdlib.servers import FTPServer
import configparser

projfolder = os.getcwd()
authorizer = DummyAuthorizer()
authorizer.add_user('user', 'password', projfolder, perm='elradfmw')

handler = FTPHandler
handler.authorizer = authorizer

configfile = os.path.join(projfolder, 'config.txt')
config = configparser.ConfigParser()
config.read(configfile)
ipadd = config.get('global', 'Ipaddress')


server = FTPServer((ipadd, 21), handler)
server.serve_forever()


```

Python Code

```python
from StorageManager import StorageManager

if __name__ == '__main__':

    manager = StorageManager() #constructor

    #FTP or Make a copy of resource files
    manager.downloadContentFromFTP() #1A - TBD
    #manager.copyContentFromLocal()   #1B

    #seperate txt and img files
    manager.categorizeimageAndTextFiles() #2
    manager.categorizeimageAndTextFiles()

    #detect anomalies/ virus in txt files
    manager.detectVirus() #3 - TBD
    manager.detectVirus()

    #archive txt and image files
    manager.compressfiles() #4

    #protect zip fciles with symmetric
    manager.protectzipfiles() #5 - TBD

    #upload zipfiles to ftp server
    #manager.uploadcontenttolocalserver() #6B
    manager.uploadcontenttoFTPserver()   #6A - TBD

#end main
```

Python Code

```python
import os
import logging
from datetime import datetime
import shutil
import configparser
import re
import zipfile
from cryptography.fernet import Fernet
from ftplib import FTP



class StorageManager:

    ##Define all variables

    #folder where all your projfiles are
    projfolder = ''

    #path for all content from ftp server
    serverfolder = ''

    #path for all content in local client
    clientfolder = ''

    #path fo all txt files
    txtfolder = ''

    #path for all img files
    imgfolder = ''

    #object that record all activities
    logger = None

    #object that parse all configuration information
    config = None


    ## 0 - Constructor to perform necessary initialization
    def __init__(self):

        self.projfolder = os.getcwd()
        self.serverfolder = os.path.join(self.projfolder, 'serverResource')
        self.virustxt = os.path.join(self.projfolder, 'virussignature.txt')
        self.configtxt = os.path.join(self.projfolder, 'config.txt')

        #a. configparser to read config file
        self.config = configparser.ConfigParser()
        self.config.read(self.configtxt)

        serverfolder = self.config.get('global', 'server_folder')
        clientfolder = self.config.get('global', 'client_folder')
```

```
            imgfolder = self.config.get('global', 'client_image_folder')
            txtfolder = self.config.get('global', 'client_text_folder')
            virussfolder = self.config.get('global', 'client_virus_folder')
            clientzipfolder = self.config.get('global', 'client_zip_folder')
            serverzipfolder = self.config.get('global', 'server_zip_folder')
            self.ipadd = self.config.get('global', 'Ipaddress')
            self.serverfolder = os.path.join(self.projfolder, serverfolder)
            self.clientfolder = os.path.join(self.projfolder, clientfolder)
            self.txtfolder = os.path.join(self.projfolder, txtfolder)
            self.imgfolder = os.path.join(self.projfolder, imgfolder)
            self.virusfolder = os.path.join(self.projfolder, virussfolder)
            self.clientzipfolder = os.path.join(self.projfolder, clientzipfolder)
            self.serverzipfolder = os.path.join(self.projfolder, serverzipfolder)

            #b. configure logging and logger object
            now = datetime.now()
            date = now.strftime('%Y%m%d')
            time = now.strftime('%H%M%S')
            logfilename = date + '_' + time + '_' + 'Ollevya.log'
            logging.basicConfig(filename=logfilename, format='%(asctime)s, %(name)s, %
(levelname)s, %(message)s',
                                filemode='w')
            self.logger = logging.getLogger()
            self.logger.setLevel(logging.INFO)


        # end __init__

        ##1A - download from ftp server to local client
        def downloadContentFromFTP(self):

            if not os.path.exists(self.clientfolder):
                os.mkdir(self.clientfolder)

            try:
                ftp = FTP()
                ftp.connect(self.ipadd, 21)
                ftp.login('user', 'password')
                # print(files)

            except ConnectionError:
                print('Connection Error')
                self.logger.info('Download from FTP server failed')

            else:
                ftp = FTP()
                ftp.connect(self.ipadd, 21)
                ftp.login('user', 'password')
                serverfolder = self.config.get('global', 'server_folder')
                downloadfrom = serverfolder
                ftp.cwd(downloadfrom)
                files = ftp.nlst()
                for file in files:
                    clientfolder = self.config.get('global', 'client_folder')
                    filepath = os.path.join(clientfolder, file)
```

```
                    ftp.retrbinary('RETR ' + file, open(filepath, 'wb').write)
                ftp.quit()
            self.logger.info('Download from {} to {}'.format(self.serverfolder, self.cl
ientfolder))
            print('downloadContentFromFTP')
        #end downloadContentFromFTP

        ##1B - Copy from local server folder to client folder
        def copyContentFromLocal(self):
            print('copyContentFromLocal')

            self.logger.info('Copy folder from local server')
            self.clientfolder = os.path.join(self.projfolder, 'clientFolder')

            if not os.path.exists(self.clientfolder):
                shutil.copytree(self.serverfolder, self.clientfolder)

        #end copyContentFromLocal

        ##2 - categorize image and text files into folders
        def categorizeimageAndTextFiles(self):
            if not os.path.exists(self.txtfolder):
                os.mkdir(self.txtfolder)

            if not os.path.exists(self.imgfolder):
                os.mkdir(self.imgfolder)

            #seperating img n txt files

            foldercontent = os.listdir(self.clientfolder)

            for files in foldercontent:
                self.srcfile = os.path.join(self.clientfolder, files)
                fname, fext = os.path.splitext(self.srcfile)

                if fext == '.txt':
                    self.destfile = os.path.join(self.txtfolder, files)
                    shutil.move(self.srcfile, self.destfile)
                else:
                    shutil.move(self.srcfile, self.imgfolder)

            self.logger.info('Moved text files into : {}'.format(self.txtfolder))
            self.logger.info('Moved image files into : {}'.format(self.imgfolder))
            print('categorizeimageAndTextFiles')
        #end categorize

        ##3 - detect files wtih virus patterns
        def detectVirus(self):
            print('detectVirus')

            if not os.path.exists(self.virusfolder):
                os.mkdir(self.virusfolder)
            else:
                print('Exist')

```

```
        #finding virus file
        textfilecontent = os.listdir(self.txtfolder)
        for line in textfilecontent:
            filepath = os.path.join(self.txtfolder, line)
            fh = open(filepath, 'r')
            filecontent = fh.readlines()
            str_line = str(filecontent)
            pattern = '\D{2}\d{6}'
            obj = re.findall(pattern, str_line)
            fh.close()

            if obj:
                self.srcfile = os.path.join(self.txtfolder, line)
                self.destfile = os.path.join(self.virusfolder, line)
                shutil.move(self.srcfile, self.destfile)

        self.logger.info('Virus file found : {}'.format(self.virusfolder))



    #end detectVirus

    ##4  - Archive txt and img files
    def compressfiles(self):
        print('compressfiles')
        if not os.path.exists(self.clientzipfolder):
            os.mkdir(self.clientzipfolder)
        else:
            print('Exist')

        self.now = datetime.now()
        time = self.now.strftime('%Y%m')
        self.ziptxtfilename = time + '-txt.zip'
        self.zipimgfilename = time + '-img.zip'
        zipfilecontent = os.listdir(self.txtfolder)
        zipfileimgcontent = os.listdir(self.imgfolder)

        #zip txt folder
        with zipfile.ZipFile(self.ziptxtfilename, 'w') as archive:
            for item in zipfilecontent:
                srctxttozip = os.path.join(self.txtfolder, item)
                archive.write(srctxttozip, arcname=item)
            self.logger.info('Zipping text files: {}'.format(self.ziptxtfilename))

        #zip img folder
        with zipfile.ZipFile(self.zipimgfilename, 'w') as archive:
            for item in zipfileimgcontent:
                srcimgtozip = os.path.join(self.imgfolder, item)
                archive.write(srcimgtozip, arcname=item)
            self.logger.info('Zipping image files: {}'.format(self.zipimgfilename))

    #end compressfiles

    ##5 - Password protect zipfiles with symmetric keys
    def protectzipfiles(self):
```

```
        print('protectzipfiles')
        key = Fernet.generate_key()
        filekey = open('filekey.key', 'wb')
        filekey.write(key)
        filekey.close()

        #use key generated for text
        self.logger.info('Encrypted text files : {}'.format(self.ziptxtfilename))
        fernet = Fernet(key)
        file = open(self.ziptxtfilename, 'rb')
        orgcontent = file.read()
        file.close()
        #print(orgcontent)
        enccontent = fernet.encrypt(orgcontent)
        encfile = open(self.ziptxtfilename, 'wb')
        encfile.write(enccontent)
        encfile.close()
        #print(enccontent)

        #use key generated for img
        self.logger.info('Encrypted text files : {}'.format(self.zipimgfilename))
        fernet = Fernet(key)
        file = open(self.zipimgfilename, 'rb')
        orgcontent = file.read()
        file.close()
        enccontent = fernet.encrypt(orgcontent)
        encfile = open(self.zipimgfilename, 'wb')
        encfile.write(enccontent)
        encfile.close()

        #move files
        shutil.move(self.ziptxtfilename, self.clientzipfolder)
        shutil.move(self.zipimgfilename, self.clientzipfolder)

    #end protectzipfiles

    ##6A - upload content to ftp server
    def uploadcontenttoFTPserver(self):
        print('uploadcontenttoFTPserver')

        if not os.path.exists(self.serverzipfolder):
            os.mkdir(self.serverzipfolder)

        try:
            ftp = FTP()
            ftp.connect(self.ipadd, 21)
            ftp.login('user', 'password')

        except ConnectionError:
            print('Connection Error')
            self.logger.info('Upload to FTP server failed')

        else:
            ftp = FTP()
            ftp.connect(self.ipadd, 21)
```

```
271                 ftp.login('user', 'password')
272                 serverclean = self.config.get('global', 'server_zip_folder')
273                 uploadto = serverclean
274                 ftp.cwd(uploadto)
275 
276                 clientclean = self.config.get('global', 'client_zip_folder')
277                 uploadfrom = clientclean
278                 files = os.listdir(clientclean)
279                 # print(files)
280 
281                 for filecontent in files:
282                     filepath = os.path.join(uploadfrom, filecontent)
283                     ftp.storbinary('STOR ' + filecontent, open(filepath, 'rb'))
284                 ftp.quit()
285             self.logger.info('Upload files from: {} to {}'.format(self.clientzipfolder,
self.serverzipfolder))
286 
287 
288         #end uploadcontent
289 
290         ##6B - Copy content to local server
291         def uploadcontenttolocalserver(self):
292             print('uploadcontenttolocalserver')
293         #end uploadcontent
294 
295 
296 
297     #end storagemanager
```