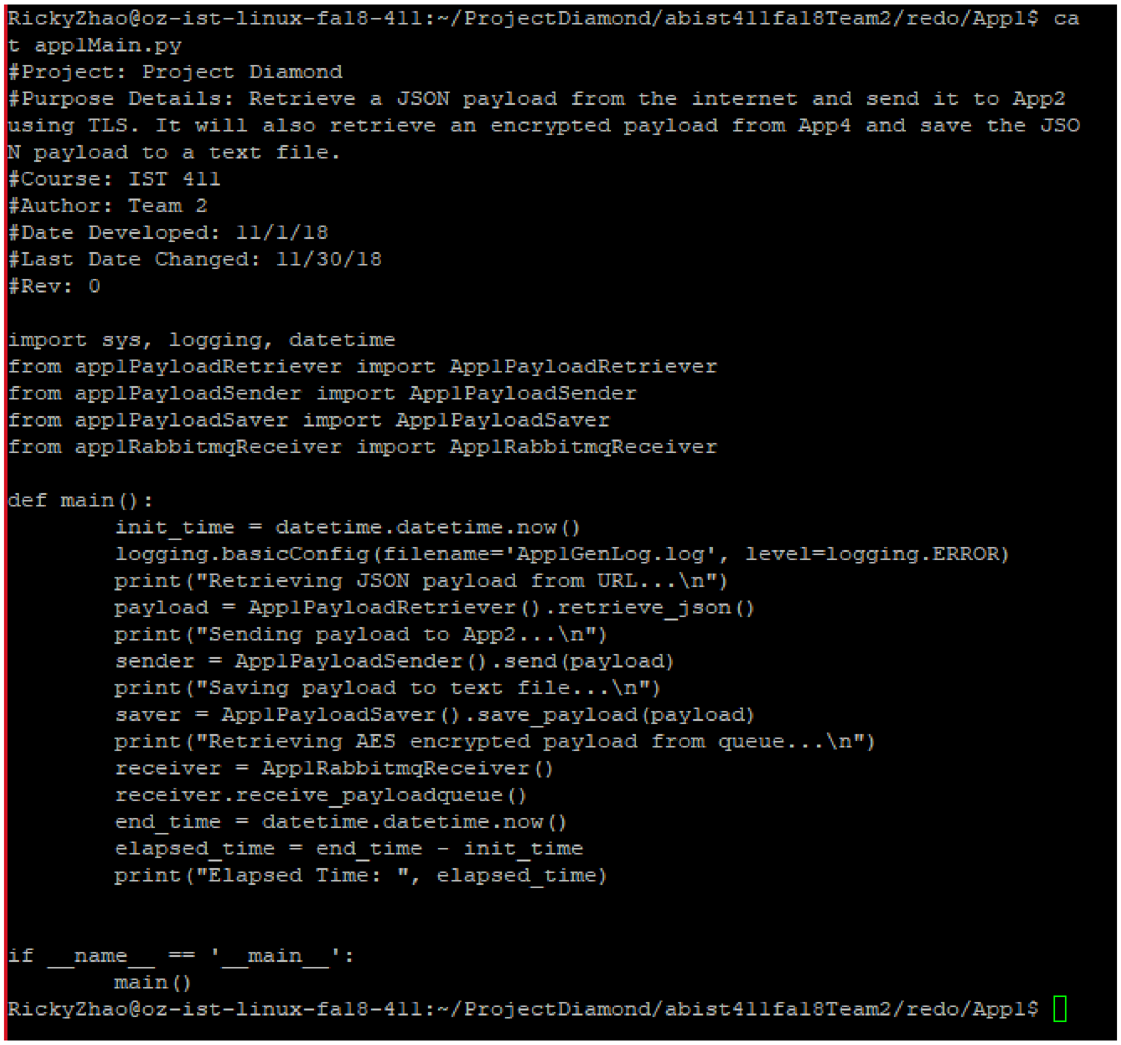
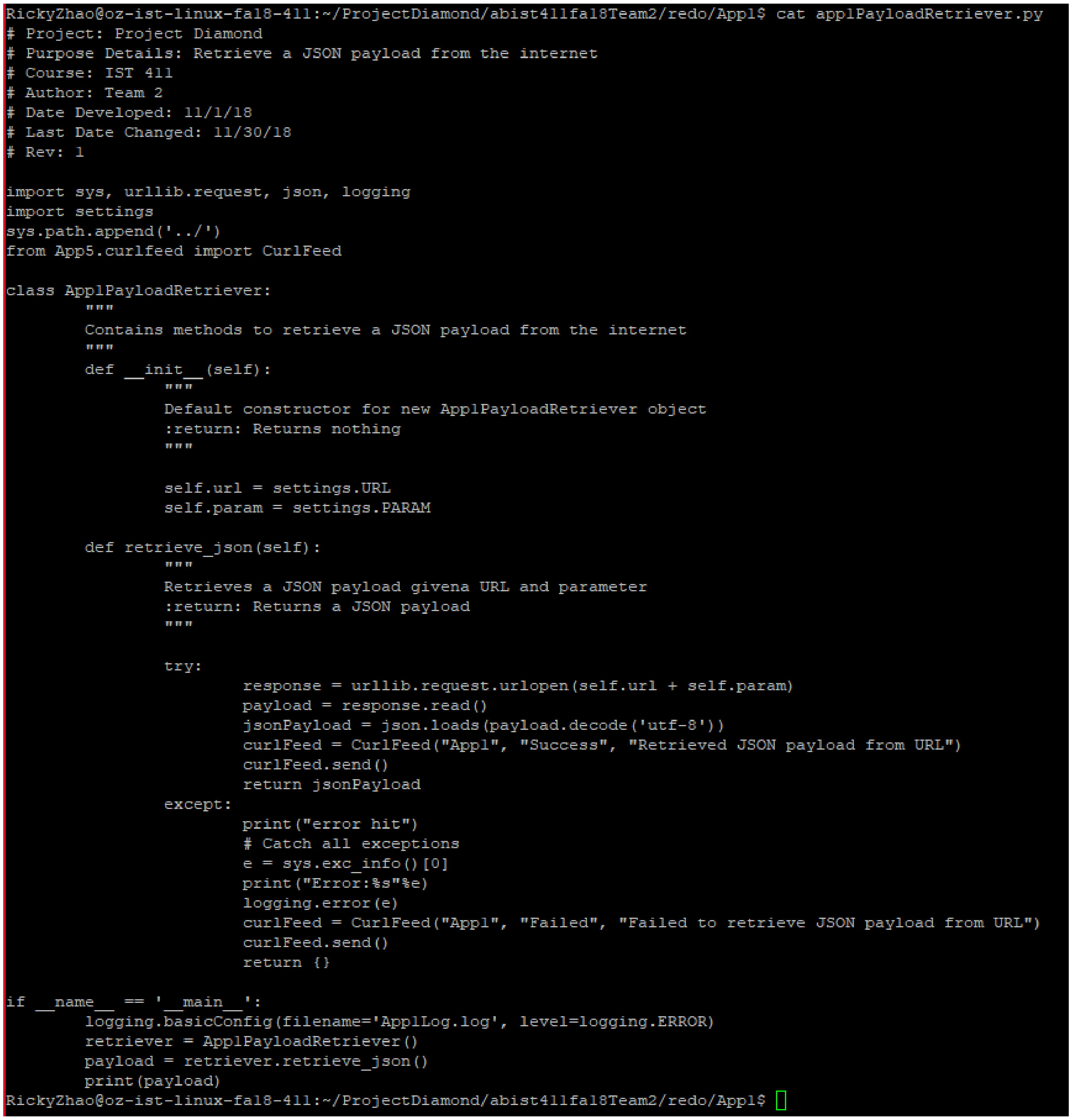
App 1

App1Main.py



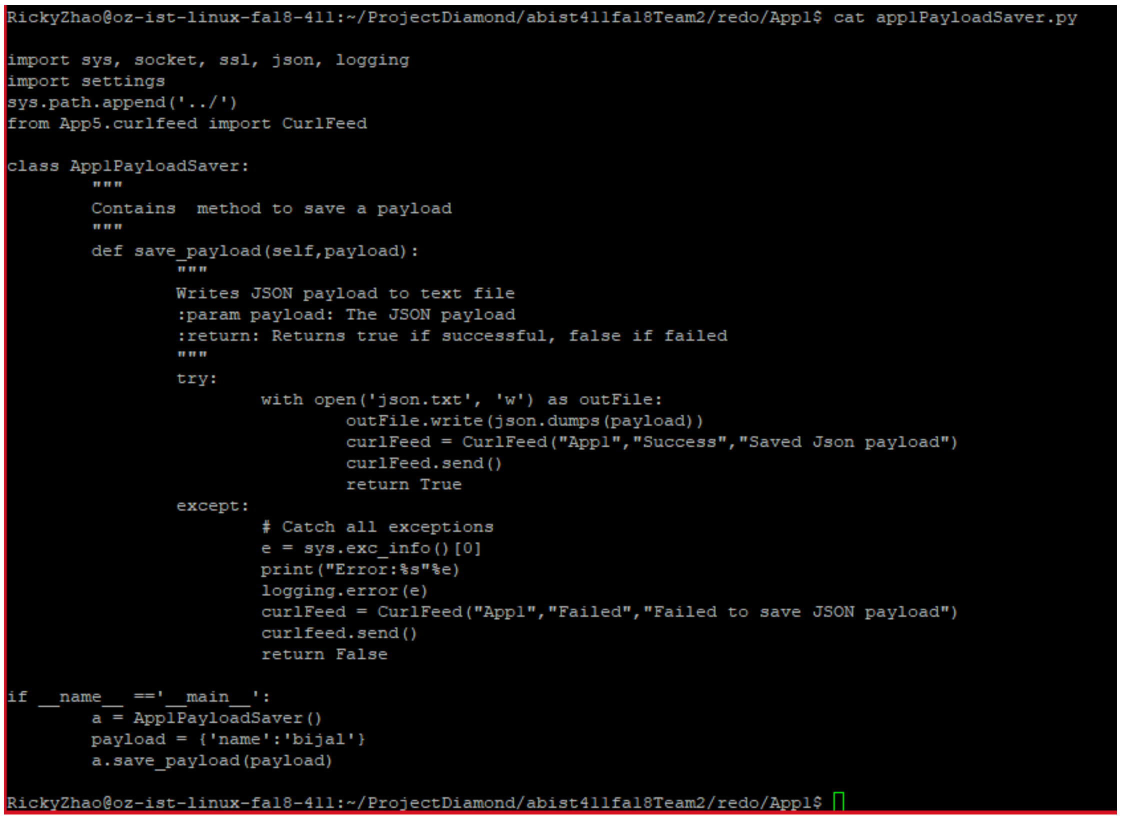
```
RickyZhao@oz-ist-linux-fa18-411:~/ProjectDiamond/abist411fa18Team2/redo/App1$ ca
t app1Main.py
#Project: Project Diamond
#Purpose Details: Retrieve a JSON payload from the internet and send it to App2
using TLS. It will also retrieve an encrypted payload from App4 and save the JSO
N payload to a text file.
#Course: IST 411
#Author: Team 2
#Date Developed: 11/1/18
#Last Date Changed: 11/30/18
#Rev: 0

import sys, logging, datetime
from app1PayloadRetriever import App1PayloadRetriever
from app1PayloadSender import App1PayloadSender
from app1PayloadSaver import App1PayloadSaver
from app1RabbitmqReceiver import App1RabbitmqReceiver

def main():
        init_time = datetime.datetime.now()
        logging.basicConfig(filename='App1GenLog.log', level=logging.ERROR)
        print("Retrieving JSON payload from URL...\n")
        payload = App1PayloadRetriever().retrieve_json()
        print("Sending payload to App2...\n")
        sender = App1PayloadSender().send(payload)
        print("Saving payload to text file...\n")
        saver = App1PayloadSaver().save_payload(payload)
        print("Retrieving AES encrypted payload from queue...\n")
        receiver = App1RabbitmqReceiver()
        receiver.receive_payloadqueue()
        end_time = datetime.datetime.now()
        elapsed_time = end_time - init_time
        print("Elapsed Time: ", elapsed_time)


if __name__ == '__main__':
        main()
RickyZhao@oz-ist-linux-fa18-411:~/ProjectDiamond/abist411fa18Team2/redo/App1$
```

App1PayloadRetriever.py



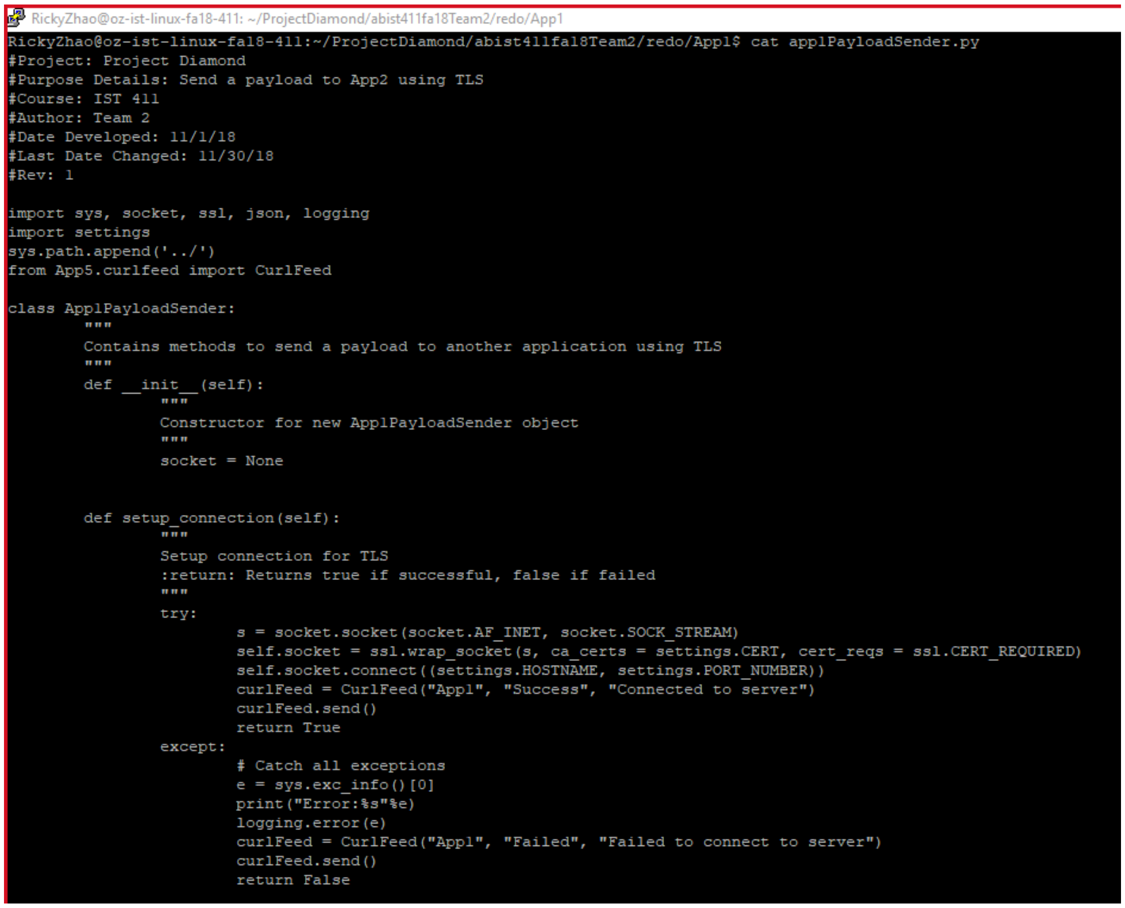
```
RickyZhao@oz-ist-linux-fa18-411:~/ProjectDiamond/abist411fa18Team2/redo/App1$ cat app1PayloadRetriever.py
# Project: Project Diamond
# Purpose Details: Retrieve a JSON payload from the internet
# Course: IST 411
# Author: Team 2
# Date Developed: 11/1/18
# Last Date Changed: 11/30/18
# Rev: 1

import sys, urllib.request, json, logging
import settings
sys.path.append('../')
from App5.curlfeed import CurlFeed

class App1PayloadRetriever:
        """
        Contains methods to retrieve a JSON payload from the internet
        """
        def __init__(self):
                """
                Default constructor for new App1PayloadRetriever object
                :return: Returns nothing
                """

                self.url = settings.URL
                self.param = settings.PARAM

        def retrieve_json(self):
                """
                Retrieves a JSON payload givena URL and parameter
                :return: Returns a JSON payload
                """

                try:
                        response = urllib.request.urlopen(self.url + self.param)
                        payload = response.read()
                        jsonPayload = json.loads(payload.decode('utf-8'))
                        curlFeed = CurlFeed("App1", "Success", "Retrieved JSON payload from URL")
                        curlFeed.send()
                        return jsonPayload
                except:
                        print("error hit")
                        # Catch all exceptions
                        e = sys.exc_info()[0]
                        print("Error:%s"%e)
                        logging.error(e)
                        curlFeed = CurlFeed("App1", "Failed", "Failed to retrieve JSON payload from URL")
                        curlFeed.send()
                        return {}

if __name__ == '__main__':
        logging.basicConfig(filename='App1Log.log', level=logging.ERROR)
        retriever = App1PayloadRetriever()
        payload = retriever.retrieve_json()
        print(payload)
RickyZhao@oz-ist-linux-fa18-411:~/ProjectDiamond/abist411fa18Team2/redo/App1$
```

App1PayloadSaver.py



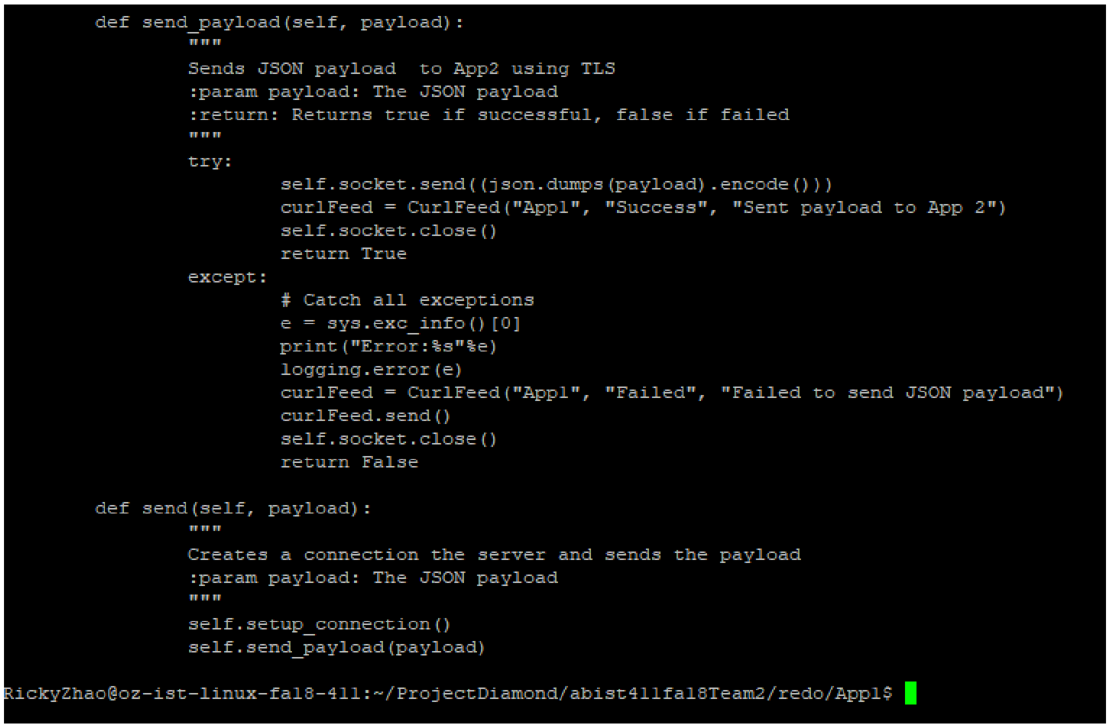
```
RickyZhao@oz-ist-linux-fa18-411:~/ProjectDiamond/abist411fa18Team2/redo/App1$ cat app1PayloadSaver.py

import sys, socket, ssl, json, logging
import settings
sys.path.append('../')
from App5.curlfeed import CurlFeed

class App1PayloadSaver:
        """
        Contains  method to save a payload
        """
        def save_payload(self,payload):
                """
                Writes JSON payload to text file
                :param payload: The JSON payload
                :return: Returns true if successful, false if failed
                """
                try:
                        with open('json.txt', 'w') as outFile:
                                outFile.write(json.dumps(payload))
                                curlFeed = CurlFeed("App1","Success","Saved Json payload")
                                curlFeed.send()
                                return True
                except:
                        # Catch all exceptions
                        e = sys.exc_info()[0]
                        print("Error:%s"%e)
                        logging.error(e)
                        curlFeed = CurlFeed("App1","Failed","Failed to save JSON payload")
                        curlfeed.send()
                        return False

if __name__ =='__main__':
        a = App1PayloadSaver()
        payload = {'name':'bijal'}
        a.save_payload(payload)

RickyZhao@oz-ist-linux-fa18-411:~/ProjectDiamond/abist411fa18Team2/redo/App1$
```

App1PayloadSender.py



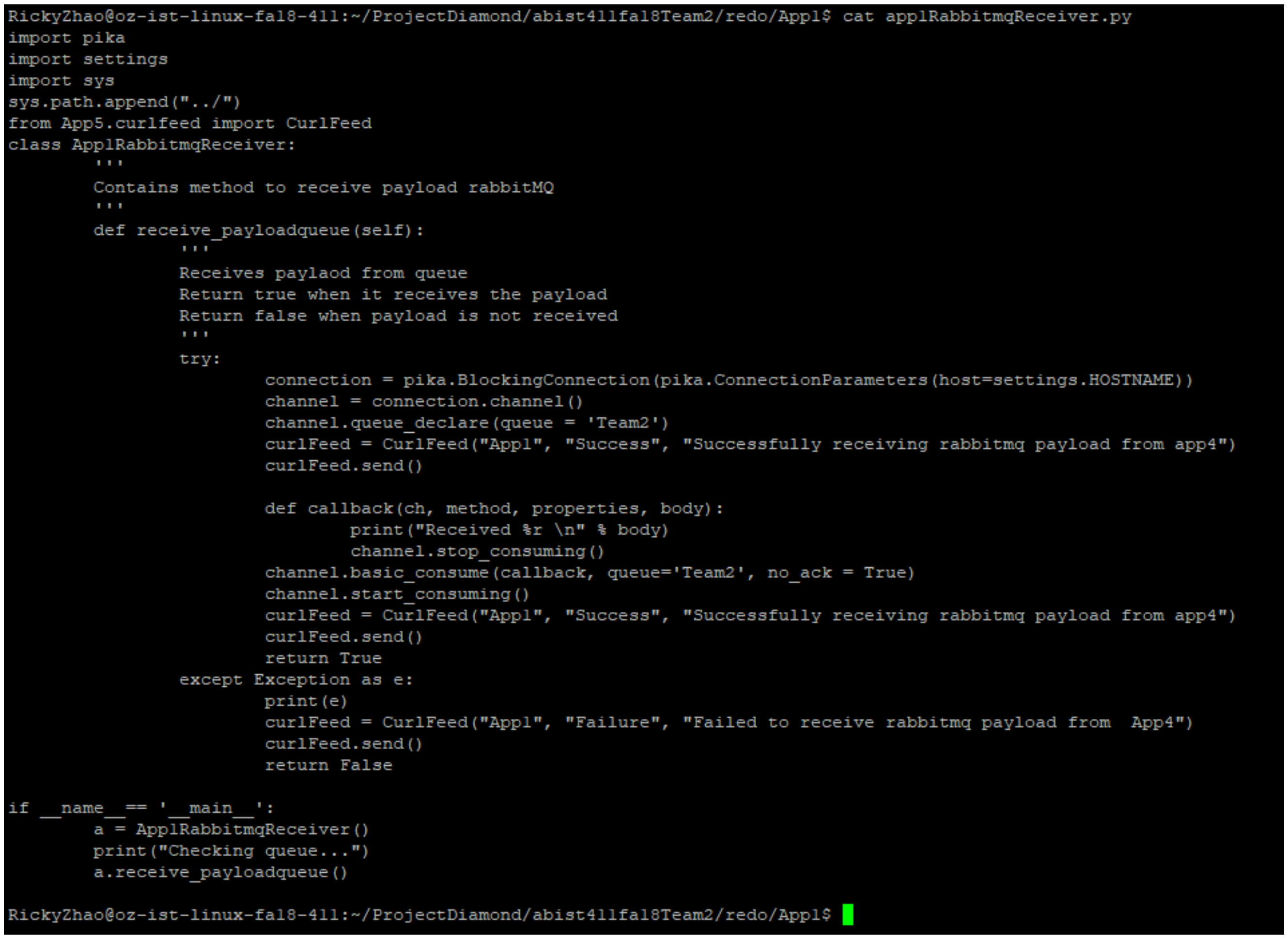
```
RickyZhao@oz-ist-linux-fa18-411: ~/ProjectDiamond/abist411fa18Team2/redo/App1
RickyZhao@oz-ist-linux-fa18-411:~/ProjectDiamond/abist411fa18Team2/redo/App1$ cat app1PayloadSender.py
#Project: Project Diamond
#Purpose Details: Send a payload to App2 using TLS
#Course: IST 411
#Author: Team 2
#Date Developed: 11/1/18
#Last Date Changed: 11/30/18
#Rev: 1

import sys, socket, ssl, json, logging
import settings
sys.path.append('../')
from App5.curlfeed import CurlFeed

class App1PayloadSender:
        """
        Contains methods to send a payload to another application using TLS
        """
        def __init__(self):
                """
                Constructor for new App1PayloadSender object
                """
                socket = None


        def setup_connection(self):
                """
                Setup connection for TLS
                :return: Returns true if successful, false if failed
                """
                try:
                        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
                        self.socket = ssl.wrap_socket(s, ca_certs = settings.CERT, cert_reqs = ssl.CERT_REQUIRED)
                        self.socket.connect((settings.HOSTNAME, settings.PORT_NUMBER))
                        curlFeed = CurlFeed("App1", "Success", "Connected to server")
                        curlFeed.send()
                        return True
                except:
                        # Catch all exceptions
                        e = sys.exc_info()[0]
                        print("Error:%s"%e)
                        logging.error(e)
                        curlFeed = CurlFeed("App1", "Failed", "Failed to connect to server")
                        curlFeed.send()
                        return False
```

App1PayloadSender.py (cont.)



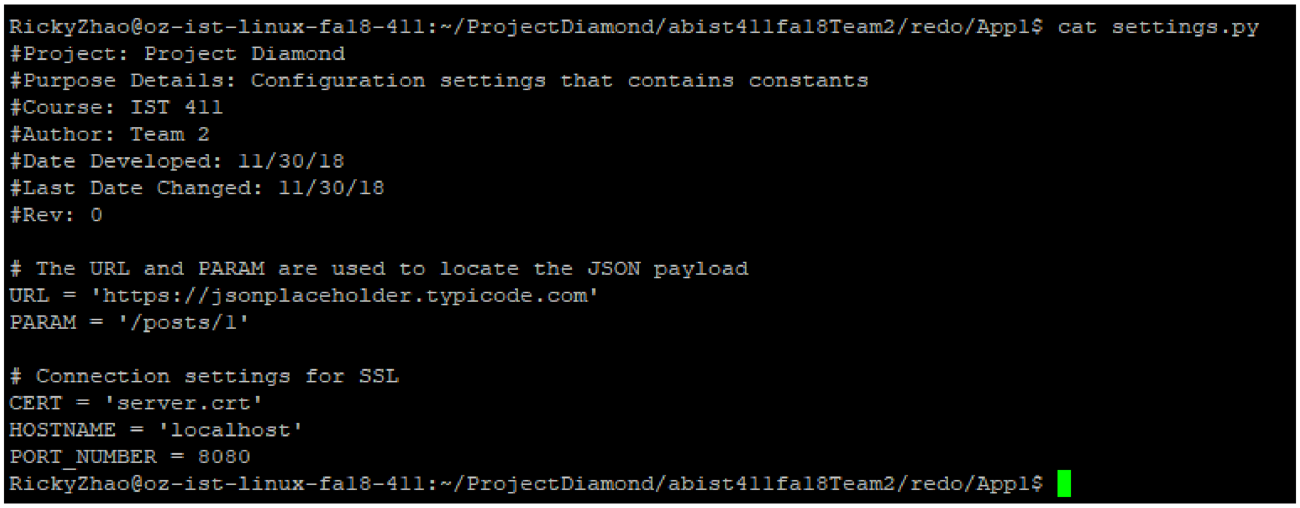
```
        def send_payload(self, payload):
                """
                Sends JSON payload  to App2 using TLS
                :param payload: The JSON payload
                :return: Returns true if successful, false if failed
                """
                try:
                        self.socket.send((json.dumps(payload).encode()))
                        curlFeed = CurlFeed("App1", "Success", "Sent payload to App 2")
                        self.socket.close()
                        return True
                except:
                        # Catch all exceptions
                        e = sys.exc_info()[0]
                        print("Error:%s"%e)
                        logging.error(e)
                        curlFeed = CurlFeed("App1", "Failed", "Failed to send JSON payload")
                        curlFeed.send()
                        self.socket.close()
                        return False

        def send(self, payload):
                """
                Creates a connection the server and sends the payload
                :param payload: The JSON payload
                """
                self.setup_connection()
                self.send_payload(payload)

RickyZhao@oz-ist-linux-fa18-411:~/ProjectDiamond/abist411fa18Team2/redo/App1$
```

App1RabbitmqReceiver.py



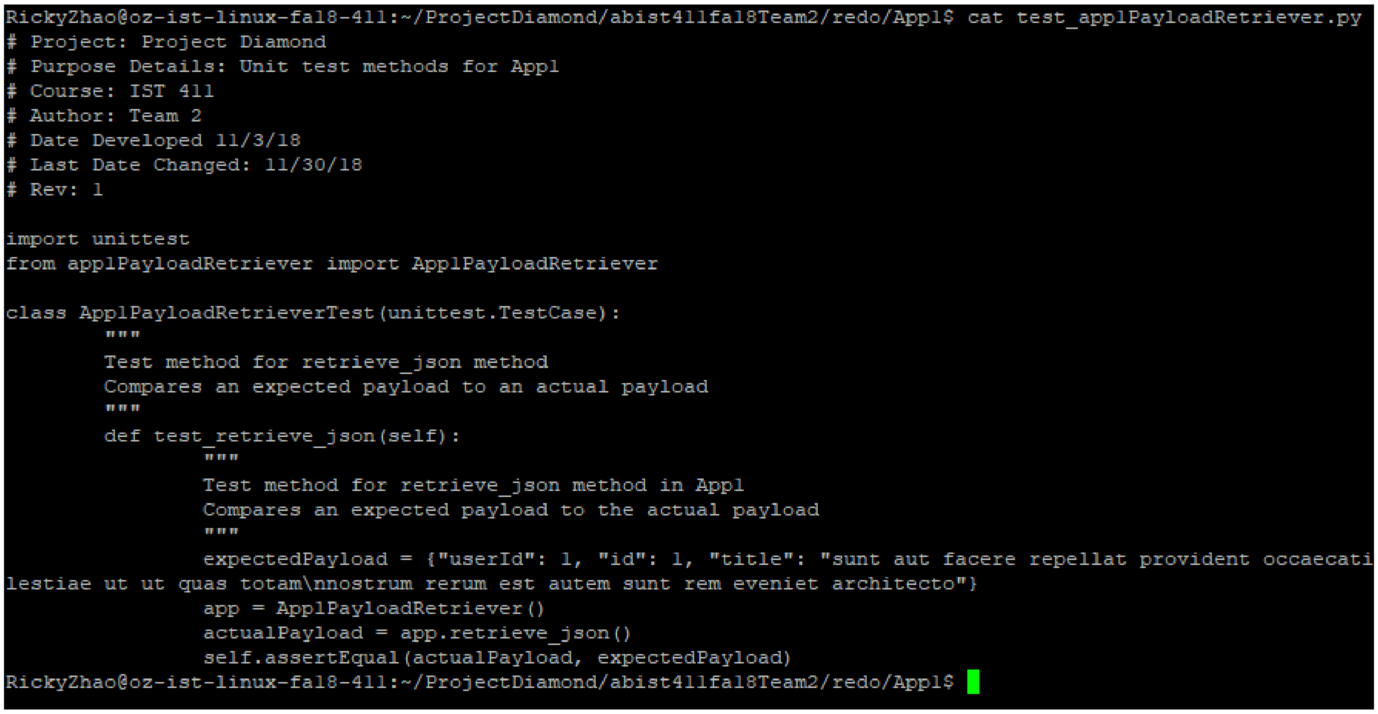
```
RickyZhao@oz-ist-linux-fa18-411:~/ProjectDiamond/abist411fa18Team2/redo/App1$ cat app1RabbitmqReceiver.py
import pika
import settings
import sys
sys.path.append("../")
from App5.curlfeed import CurlFeed
class App1RabbitmqReceiver:
        '''
        Contains method to receive payload rabbitMQ
        '''
        def receive_payloadqueue(self):
                '''
                Receives paylaod from queue
                Return true when it receives the payload
                Return false when payload is not received
                '''
                try:
                        connection = pika.BlockingConnection(pika.ConnectionParameters(host=settings.HOSTNAME))
                        channel = connection.channel()
                        channel.queue_declare(queue = 'Team2')
                        curlFeed = CurlFeed("App1", "Success", "Successfully receiving rabbitmq payload from app4")
                        curlFeed.send()

                        def callback(ch, method, properties, body):
                                print("Received %r \n" % body)
                                channel.stop_consuming()
                        channel.basic_consume(callback, queue='Team2', no_ack = True)
                        channel.start_consuming()
                        curlFeed = CurlFeed("App1", "Success", "Successfully receiving rabbitmq payload from app4")
                        curlFeed.send()
                        return True
                except Exception as e:
                        print(e)
                        curlFeed = CurlFeed("App1", "Failure", "Failed to receive rabbitmq payload from  App4")
                        curlFeed.send()
                        return False

if __name__== '__main__':
        a = App1RabbitmqReceiver()
        print("Checking queue...")
        a.receive_payloadqueue()

RickyZhao@oz-ist-linux-fa18-411:~/ProjectDiamond/abist411fa18Team2/redo/App1$
```

Settings.py



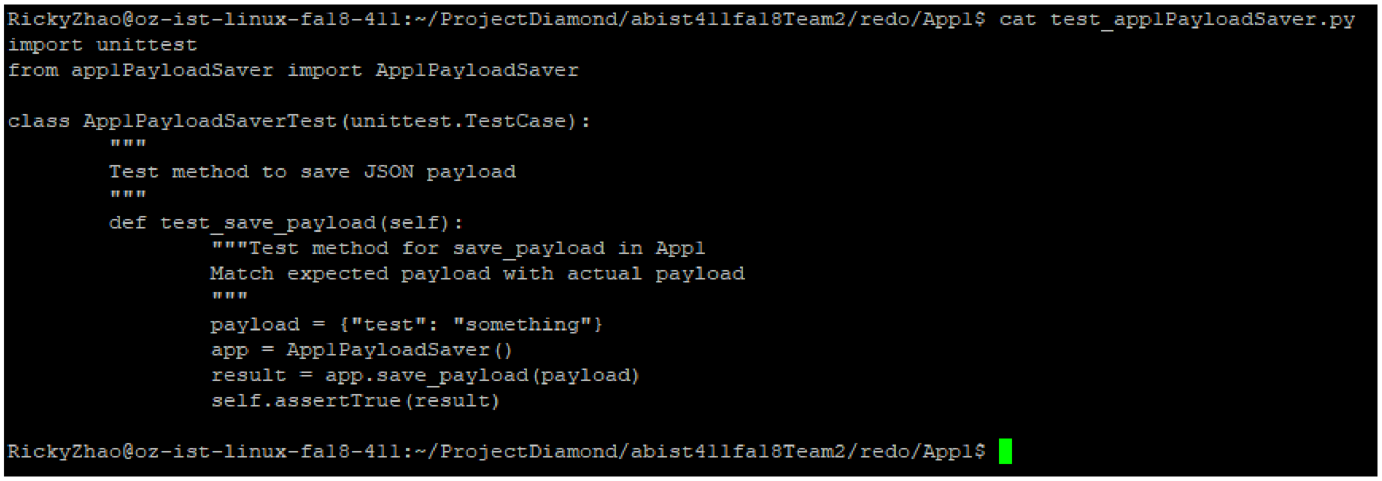
```
RickyZhao@oz-ist-linux-fa18-411:~/ProjectDiamond/abist411fa18Team2/redo/App1$ cat settings.py
#Project: Project Diamond
#Purpose Details: Configuration settings that contains constants
#Course: IST 411
#Author: Team 2
#Date Developed: 11/30/18
#Last Date Changed: 11/30/18
#Rev: 0

# The URL and PARAM are used to locate the JSON payload
URL = 'https://jsonplaceholder.typicode.com'
PARAM = '/posts/1'

# Connection settings for SSL
CERT = 'server.crt'
HOSTNAME = 'localhost'
PORT_NUMBER = 8080
RickyZhao@oz-ist-linux-fa18-411:~/ProjectDiamond/abist411fa18Team2/redo/App1$
```

Test_app1PayloadRetriever.py



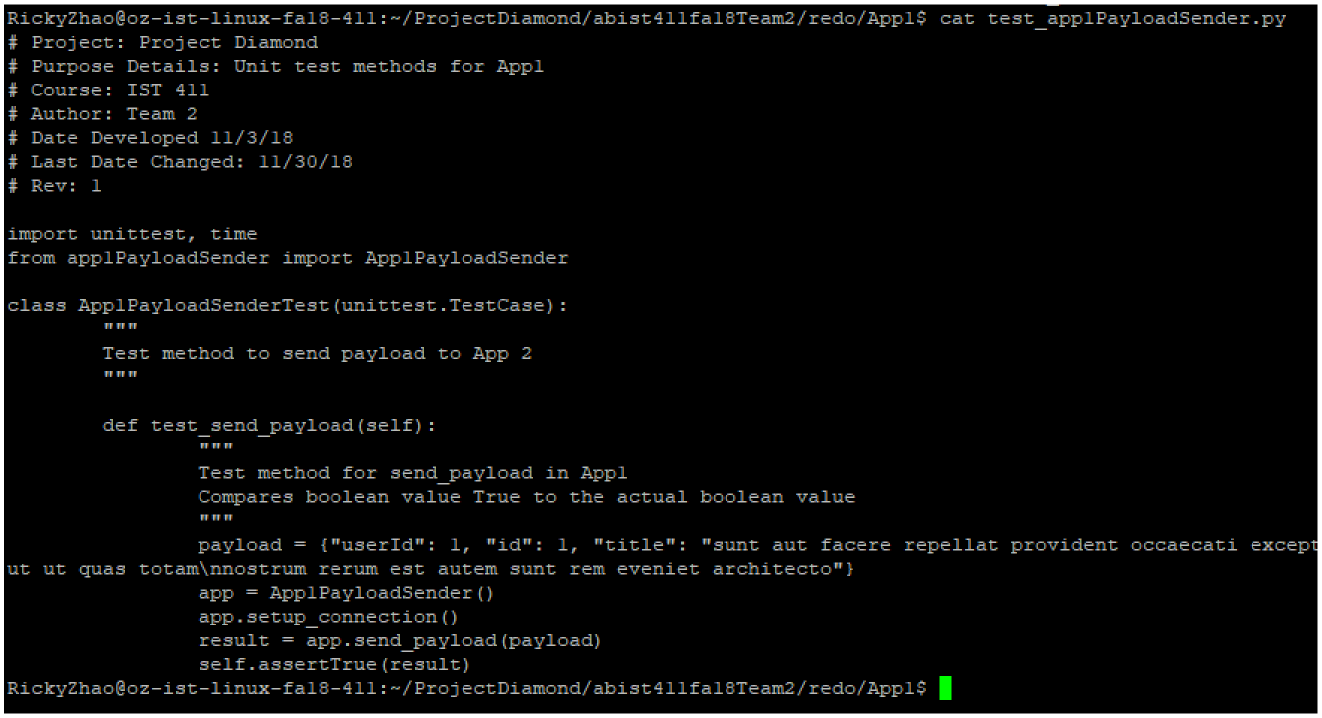
```
RickyZhao@oz-ist-linux-fa18-411:~/ProjectDiamond/abist411fa18Team2/redo/App1$ cat test_app1PayloadRetriever.py
# Project: Project Diamond
# Purpose Details: Unit test methods for App1
# Course: IST 411
# Author: Team 2
# Date Developed 11/3/18
# Last Date Changed: 11/30/18
# Rev: 1

import unittest
from app1PayloadRetriever import App1PayloadRetriever

class App1PayloadRetrieverTest(unittest.TestCase):
        """
        Test method for retrieve_json method
        Compares an expected payload to an actual payload
        """
        def test_retrieve_json(self):
                """
                Test method for retrieve_json method in App1
                Compares an expected payload to the actual payload
                """
                expectedPayload = {"userId": 1, "id": 1, "title": "sunt aut facere repellat provident occaecati
lestiae ut ut quas totam\nnostrum rerum est autem sunt rem eveniet architecto"}
                app = App1PayloadRetriever()
                actualPayload = app.retrieve_json()
                self.assertEqual(actualPayload, expectedPayload)
RickyZhao@oz-ist-linux-fa18-411:~/ProjectDiamond/abist411fa18Team2/redo/App1$
```

Test_app1PayloadSaver.py



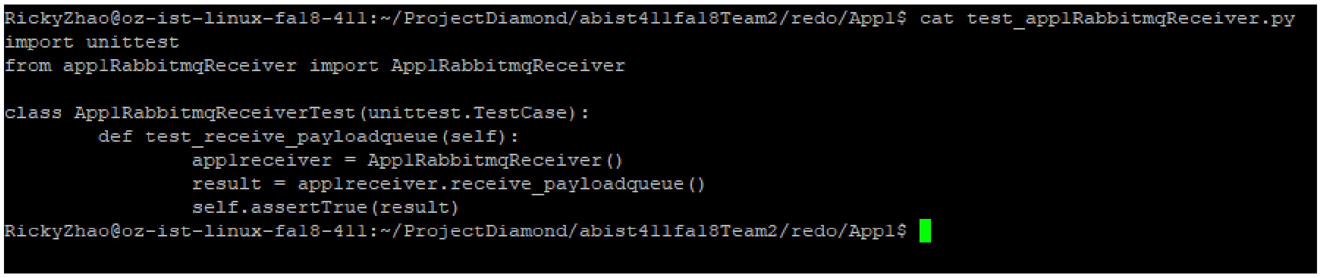
```
RickyZhao@oz-ist-linux-fa18-411:~/ProjectDiamond/abist411fa18Team2/redo/App1$ cat test_app1PayloadSaver.py
import unittest
from app1PayloadSaver import App1PayloadSaver

class App1PayloadSaverTest(unittest.TestCase):
        """
        Test method to save JSON payload
        """
        def test_save_payload(self):
                """Test method for save_payload in App1
                Match expected payload with actual payload
                """
                payload = {"test": "something"}
                app = App1PayloadSaver()
                result = app.save_payload(payload)
                self.assertTrue(result)

RickyZhao@oz-ist-linux-fa18-411:~/ProjectDiamond/abist411fa18Team2/redo/App1$ 
```

Test_app1PayloadSender.py



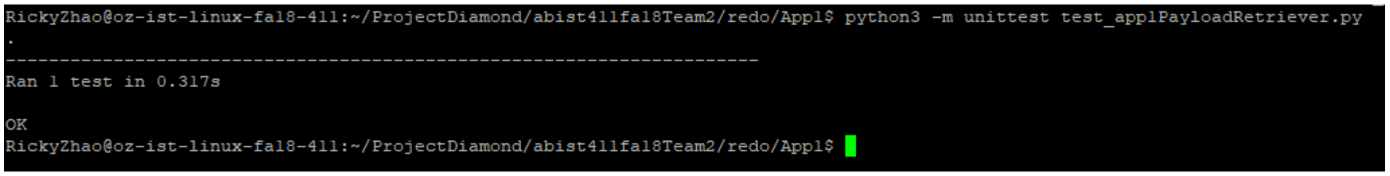
```
RickyZhao@oz-ist-linux-fa18-411:~/ProjectDiamond/abist411fa18Team2/redo/App1$ cat test_app1PayloadSender.py
# Project: Project Diamond
# Purpose Details: Unit test methods for App1
# Course: IST 411
# Author: Team 2
# Date Developed 11/3/18
# Last Date Changed: 11/30/18
# Rev: 1

import unittest, time
from app1PayloadSender import App1PayloadSender

class App1PayloadSenderTest(unittest.TestCase):
        """
        Test method to send payload to App 2
        """

        def test_send_payload(self):
                """
                Test method for send_payload in App1
                Compares boolean value True to the actual boolean value
                """
                payload = {"userId": 1, "id": 1, "title": "sunt aut facere repellat provident occaecati except
ut ut quas totam\nnostrum rerum est autem sunt rem eveniet architecto"}
                app = App1PayloadSender()
                app.setup_connection()
                result = app.send_payload(payload)
                self.assertTrue(result)
RickyZhao@oz-ist-linux-fa18-411:~/ProjectDiamond/abist411fa18Team2/redo/App1$ 
```

Test_app1RabbitmqReceiver.py



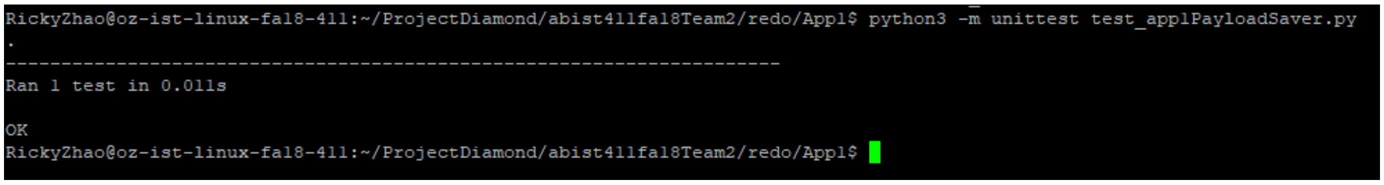
```
RickyZhao@oz-ist-linux-fa18-411:~/ProjectDiamond/abist411fa18Team2/redo/App1$ cat test_app1RabbitmqReceiver.py
import unittest
from app1RabbitmqReceiver import App1RabbitmqReceiver

class App1RabbitmqReceiverTest(unittest.TestCase):
        def test_receive_payloadqueue(self):
                app1receiver = App1RabbitmqReceiver()
                result = app1receiver.receive_payloadqueue()
                self.assertTrue(result)
RickyZhao@oz-ist-linux-fa18-411:~/ProjectDiamond/abist411fa18Team2/redo/App1$
```

Test_app1PayloadRetriever.py running



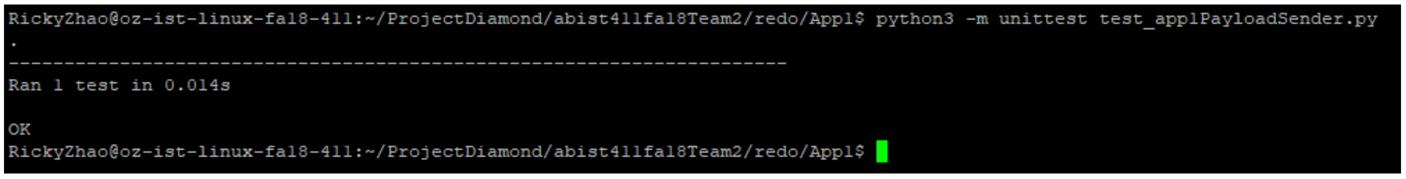
```
RickyZhao@oz-ist-linux-fa18-411:~/ProjectDiamond/abist411fa18Team2/redo/App1$ python3 -m unittest test_app1PayloadRetriever.py
.
----------------------------------------------------------------------
Ran 1 test in 0.317s

OK
RickyZhao@oz-ist-linux-fa18-411:~/ProjectDiamond/abist411fa18Team2/redo/App1$
```

Test_app1PayloadSaver.py running



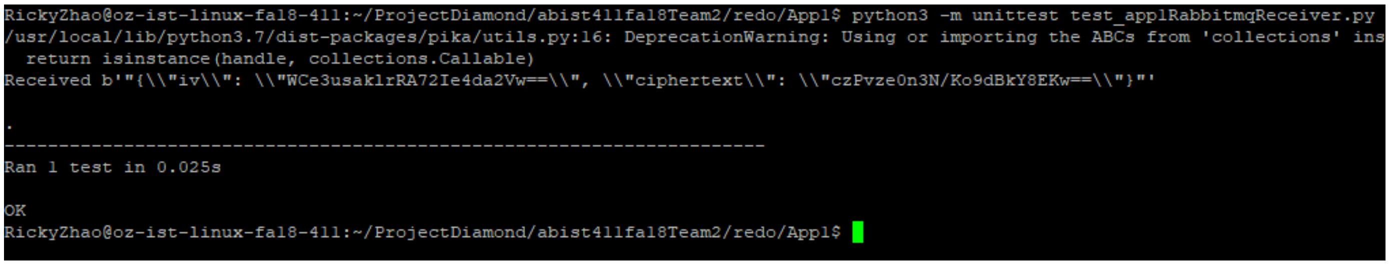
```
RickyZhao@oz-ist-linux-fa18-411:~/ProjectDiamond/abist411fa18Team2/redo/App1$ python3 -m unittest test_app1PayloadSaver.py
.
----------------------------------------------------------------------
Ran 1 test in 0.011s

OK
RickyZhao@oz-ist-linux-fa18-411:~/ProjectDiamond/abist411fa18Team2/redo/App1$
```

Test_app1PayloadSender.py running



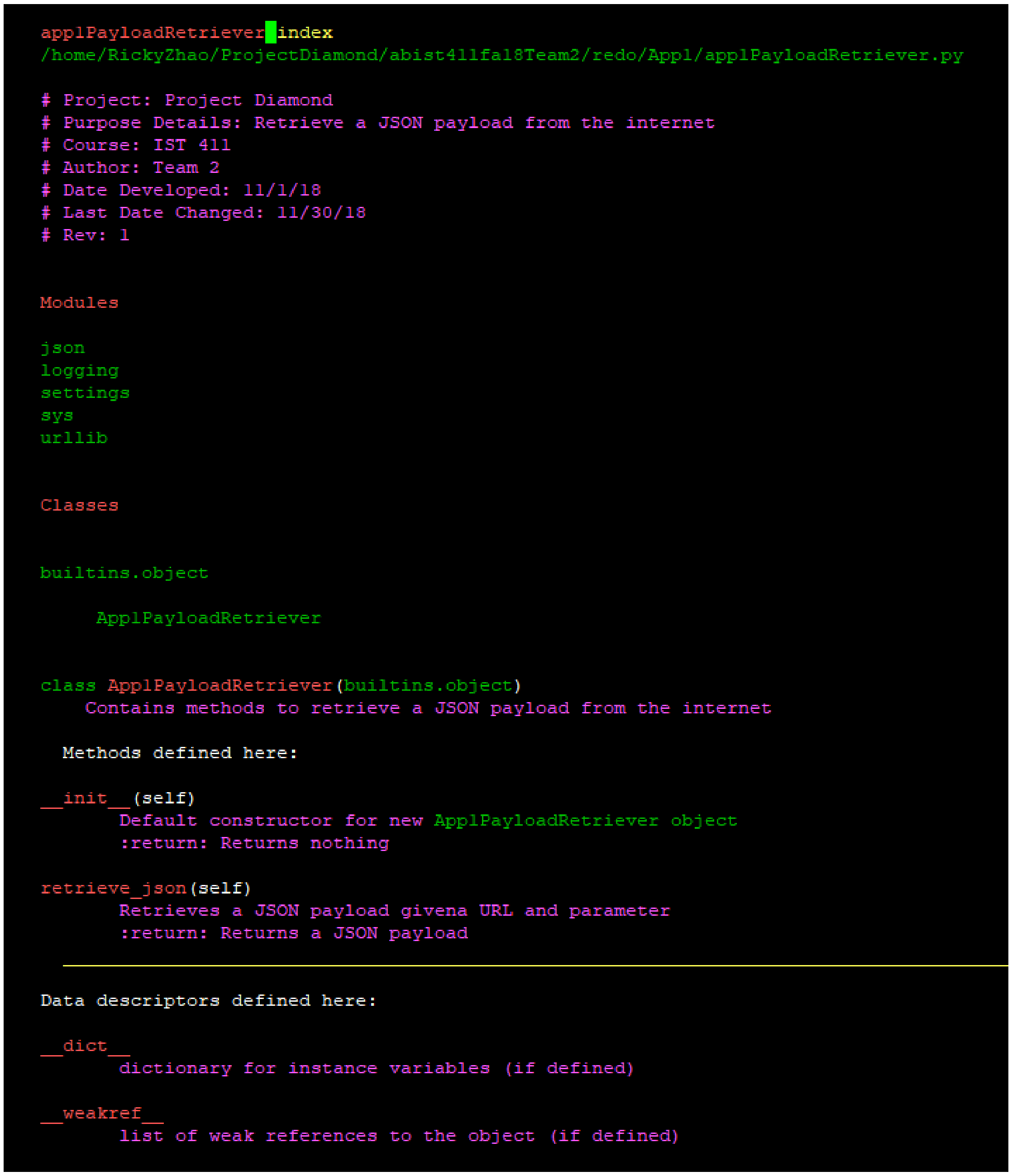
```
RickyZhao@oz-ist-linux-fa18-411:~/ProjectDiamond/abist411fa18Team2/redo/App1$ python3 -m unittest test_app1PayloadSender.py
.
----------------------------------------------------------------------
Ran 1 test in 0.014s

OK
RickyZhao@oz-ist-linux-fa18-411:~/ProjectDiamond/abist411fa18Team2/redo/App1$
```

Test_app1RabbitmqReceiver.py running



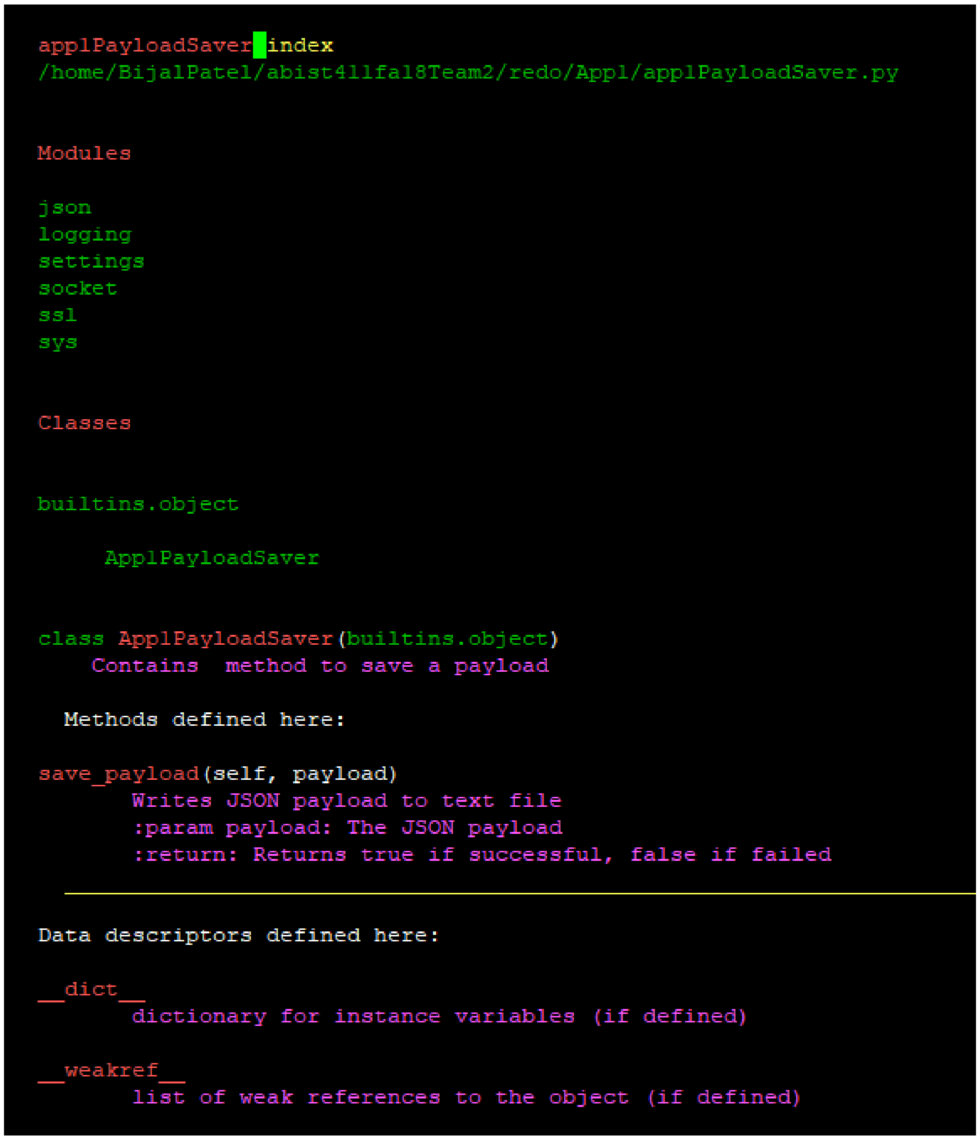
```
RickyZhao@oz-ist-linux-fa18-411:~/ProjectDiamond/abist411fa18Team2/redo/App1$ python3 -m unittest test_app1RabbitmqReceiver.py
/usr/local/lib/python3.7/dist-packages/pika/utils.py:16: DeprecationWarning: Using or importing the ABCs from 'collections' ins
  return isinstance(handle, collections.Callable)
Received b'"{\\"iv\\": \\"WCe3usaklrRA72Ie4da2Vw==\\", \\"ciphertext\\": \\"czPvze0n3N/Ko9dBkY8EKw==\\"}"'

.
----------------------------------------------------------------------
Ran 1 test in 0.025s

OK
RickyZhao@oz-ist-linux-fa18-411:~/ProjectDiamond/abist411fa18Team2/redo/App1$
```

App1PayloadRetriever.html



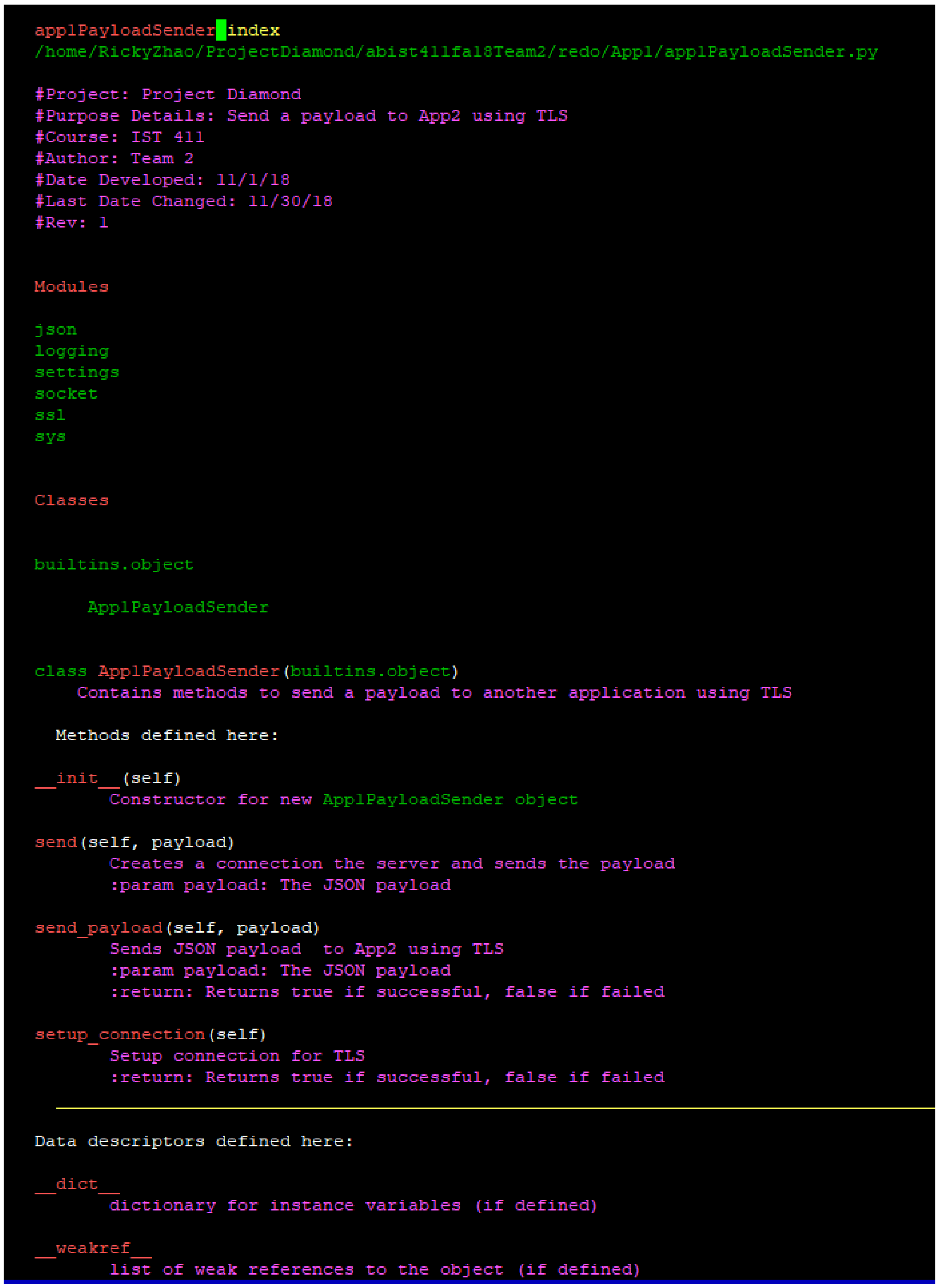
```
app1PayloadRetriever index
/home/RickyZhao/ProjectDiamond/abist411fal8Team2/redo/App1/app1PayloadRetriever.py

# Project: Project Diamond
# Purpose Details: Retrieve a JSON payload from the internet
# Course: IST 411
# Author: Team 2
# Date Developed: 11/1/18
# Last Date Changed: 11/30/18
# Rev: 1


Modules

json
logging
settings
sys
urllib


Classes


builtins.object

     App1PayloadRetriever


class App1PayloadRetriever(builtins.object)
    Contains methods to retrieve a JSON payload from the internet

  Methods defined here:

__init__(self)
       Default constructor for new App1PayloadRetriever object
       :return: Returns nothing

retrieve_json(self)
       Retrieves a JSON payload givena URL and parameter
       :return: Returns a JSON payload

Data descriptors defined here:

__dict__
       dictionary for instance variables (if defined)

__weakref__
       list of weak references to the object (if defined)
```

App1PayloadSaver.html



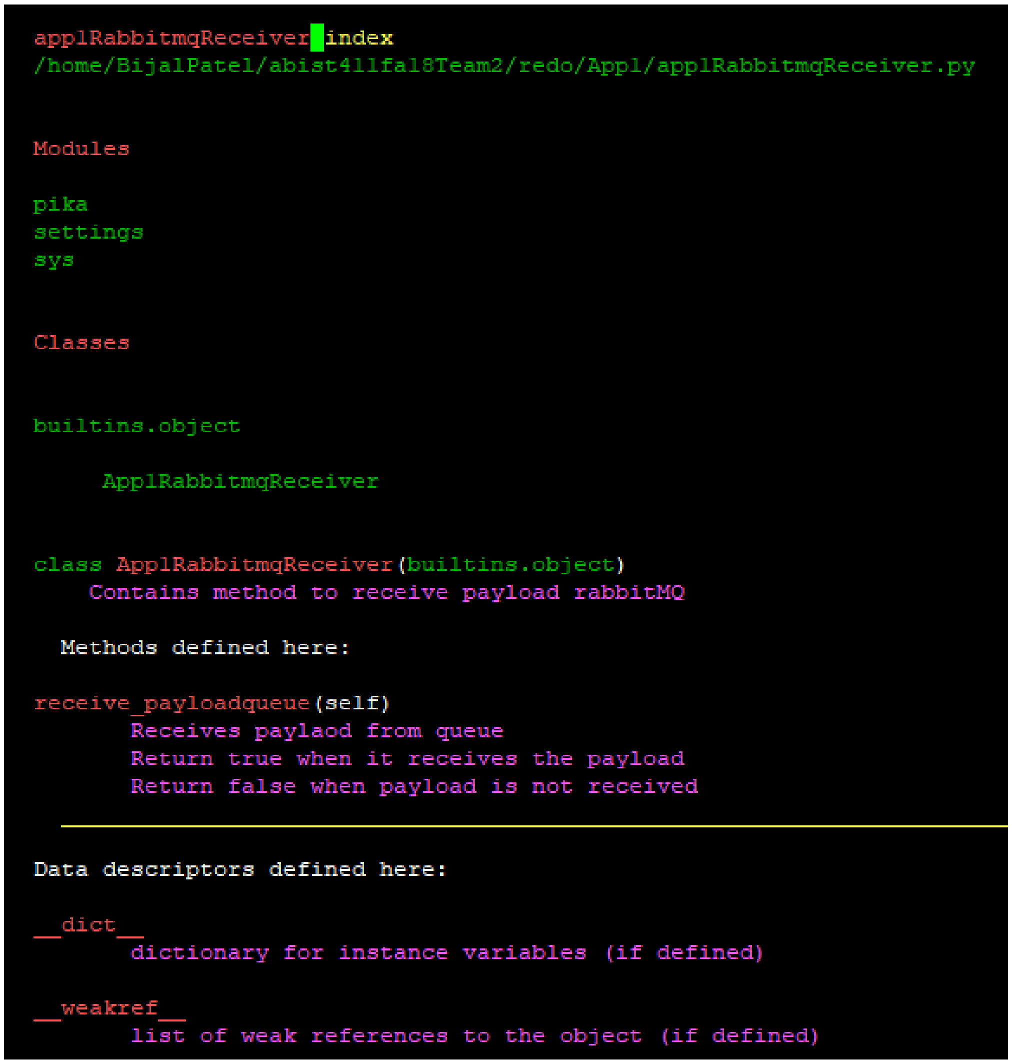
```
app1PayloadSaver index
/home/BijalPatel/abist411fa18Team2/redo/App1/app1PayloadSaver.py


Modules

json
logging
settings
socket
ssl
sys


Classes


builtins.object

     App1PayloadSaver


class App1PayloadSaver(builtins.object)
    Contains  method to save a payload

  Methods defined here:

save_payload(self, payload)
       Writes JSON payload to text file
       :param payload: The JSON payload
       :return: Returns true if successful, false if failed

Data descriptors defined here:

__dict__
       dictionary for instance variables (if defined)

__weakref__
       list of weak references to the object (if defined)
```

App1PayloadSender.html



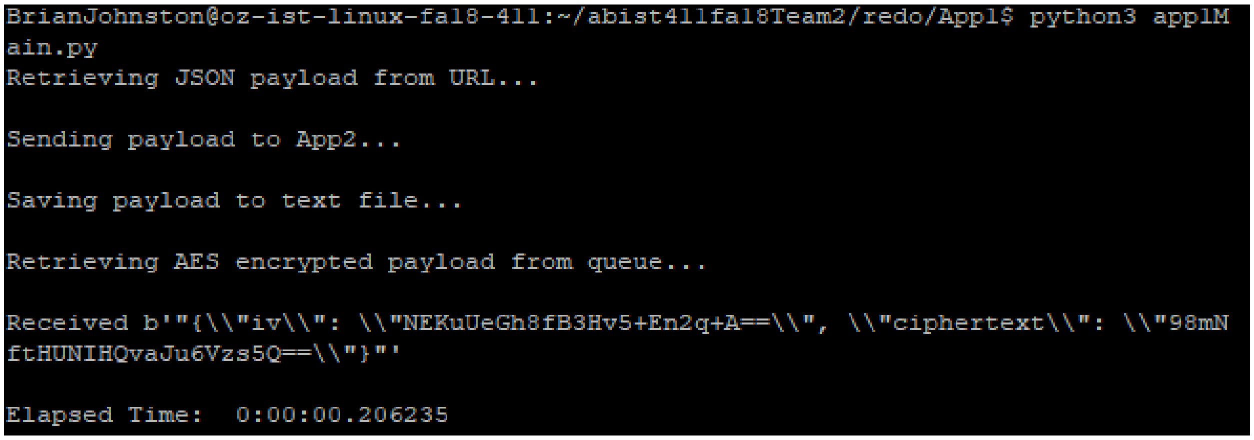
```
app1PayloadSender index
/home/RickyZhao/ProjectDiamond/abist411fal8Team2/redo/App1/app1PayloadSender.py

#Project: Project Diamond
#Purpose Details: Send a payload to App2 using TLS
#Course: IST 411
#Author: Team 2
#Date Developed: 11/1/18
#Last Date Changed: 11/30/18
#Rev: 1


Modules

json
logging
settings
socket
ssl
sys


Classes


builtins.object

     App1PayloadSender


class App1PayloadSender(builtins.object)
    Contains methods to send a payload to another application using TLS

  Methods defined here:

__init__(self)
       Constructor for new App1PayloadSender object

send(self, payload)
       Creates a connection the server and sends the payload
       :param payload: The JSON payload

send_payload(self, payload)
       Sends JSON payload  to App2 using TLS
       :param payload: The JSON payload
       :return: Returns true if successful, false if failed

setup_connection(self)
       Setup connection for TLS
       :return: Returns true if successful, false if failed

Data descriptors defined here:

__dict__
       dictionary for instance variables (if defined)

__weakref__
       list of weak references to the object (if defined)
```

App1RabbitmqReceiver.html



```
app1RabbitmqReceiver index
/home/BijalPatel/abist411fal8Team2/redo/App1/app1RabbitmqReceiver.py


Modules

pika
settings
sys


Classes


builtins.object

     App1RabbitmqReceiver


class App1RabbitmqReceiver(builtins.object)
    Contains method to receive payload rabbitMQ

  Methods defined here:

receive_payloadqueue(self)
       Receives paylaod from queue
       Return true when it receives the payload
       Return false when payload is not received

  ______________________________________________________________________

Data descriptors defined here:

__dict__
       dictionary for instance variables (if defined)

__weakref__
       list of weak references to the object (if defined)
```

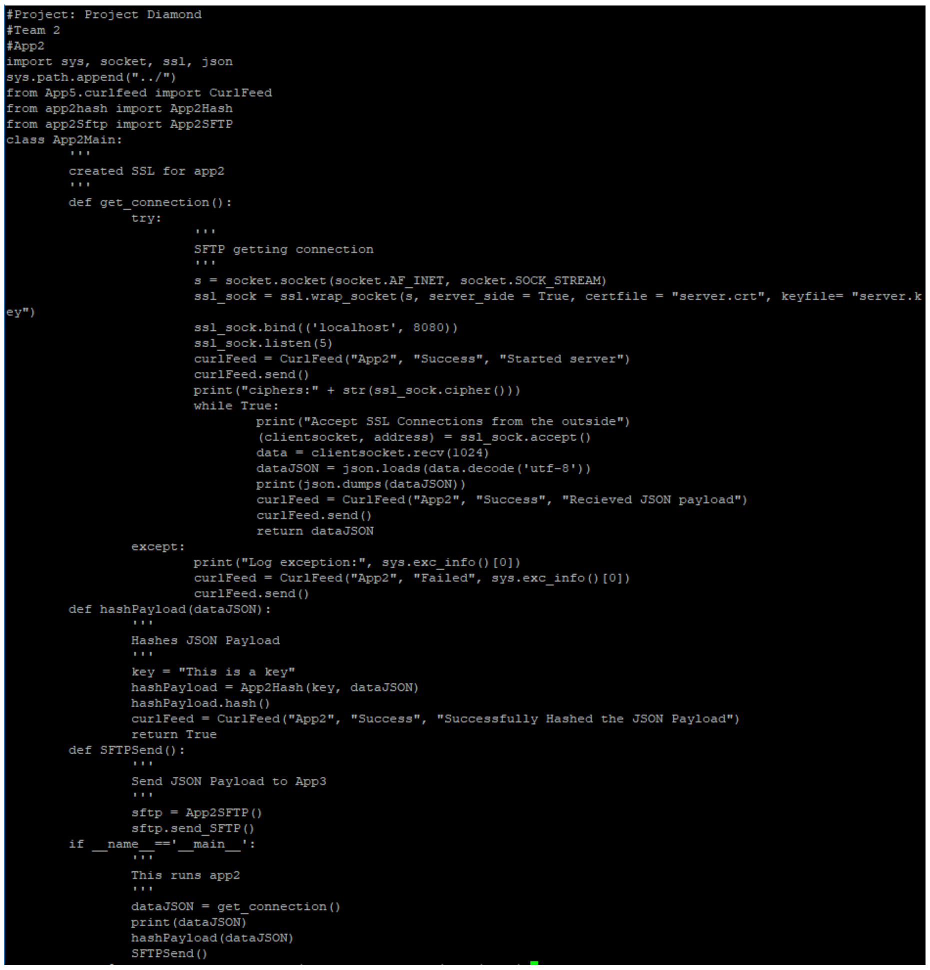
App1Main.py running



```
BrianJohnston@oz-ist-linux-fa18-411:~/abist411fa18Team2/redo/App1$ python3 app1M
ain.py
Retrieving JSON payload from URL...

Sending payload to App2...

Saving payload to text file...

Retrieving AES encrypted payload from queue...

Received b'"{\\"iv\\": \\"NEKuUeGh8fB3Hv5+En2q+A==\\", \\"ciphertext\\": \\"98mN
ftHUNIHQvaJu6Vzs5Q==\\"}"'

Elapsed Time:  0:00:00.206235
```

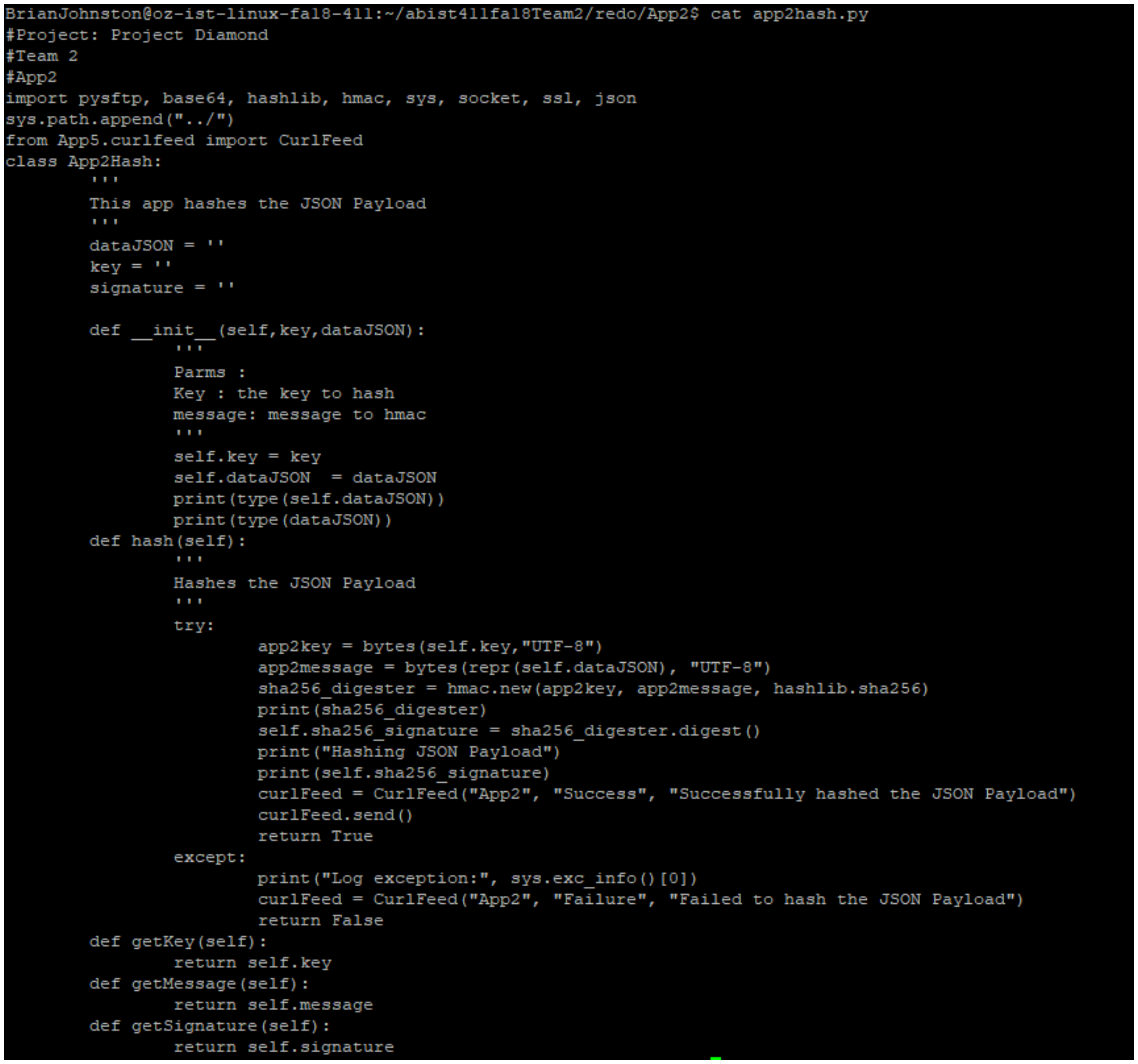
App 2

App2Main



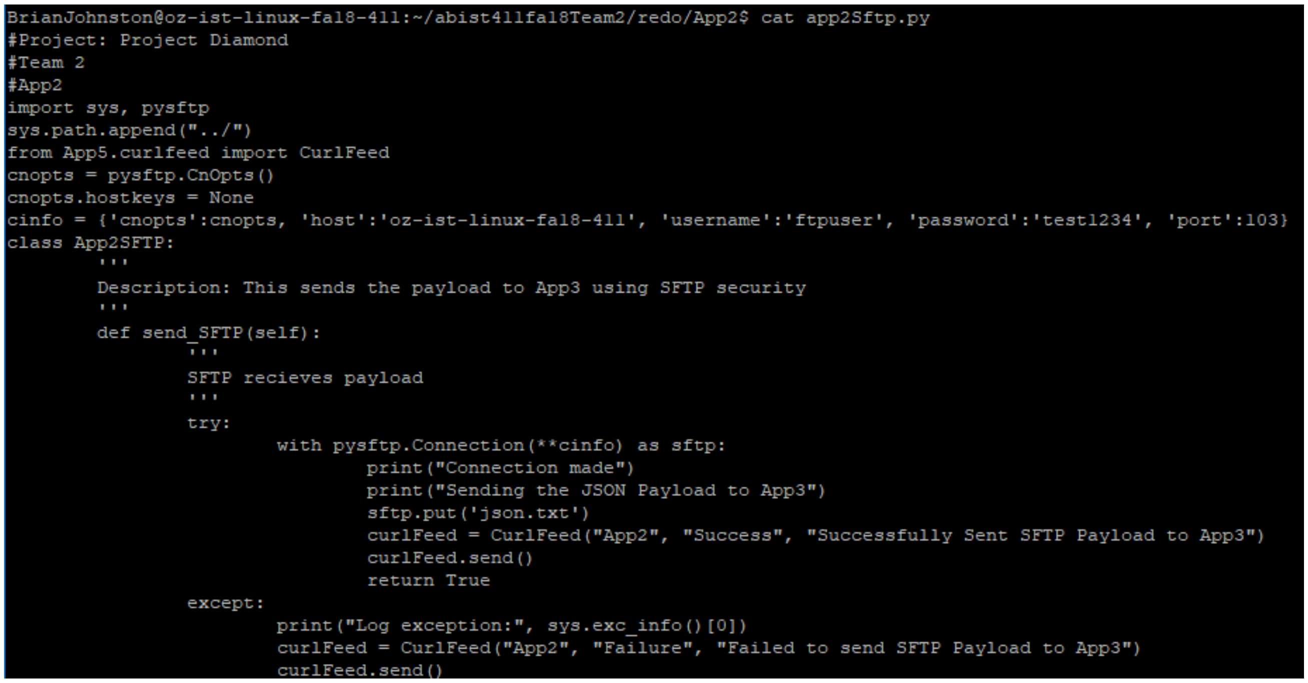
```
#Project: Project Diamond
#Team 2
#App2
import sys, socket, ssl, json
sys.path.append("../")
from App5.curlfeed import CurlFeed
from app2hash import App2Hash
from app2Sftp import App2SFTP
class App2Main:
        '''
        created SSL for app2
        '''
        def get_connection():
                try:
                        '''
                        SFTP getting connection
                        '''
                        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
                        ssl_sock = ssl.wrap_socket(s, server_side = True, certfile = "server.crt", keyfile= "server.k
ey")
                        ssl_sock.bind(('localhost', 8080))
                        ssl_sock.listen(5)
                        curlFeed = CurlFeed("App2", "Success", "Started server")
                        curlFeed.send()
                        print("ciphers:" + str(ssl_sock.cipher()))
                        while True:
                                print("Accept SSL Connections from the outside")
                                (clientsocket, address) = ssl_sock.accept()
                                data = clientsocket.recv(1024)
                                dataJSON = json.loads(data.decode('utf-8'))
                                print(json.dumps(dataJSON))
                                curlFeed = CurlFeed("App2", "Success", "Recieved JSON payload")
                                curlFeed.send()
                                return dataJSON
                except:
                        print("Log exception:", sys.exc_info()[0])
                        curlFeed = CurlFeed("App2", "Failed", sys.exc_info()[0])
                        curlFeed.send()
        def hashPayload(dataJSON):
                '''
                Hashes JSON Payload
                '''
                key = "This is a key"
                hashPayload = App2Hash(key, dataJSON)
                hashPayload.hash()
                curlFeed = CurlFeed("App2", "Success", "Successfully Hashed the JSON Payload")
                return True
        def SFTPSend():
                '''
                Send JSON Payload to App3
                '''
                sftp = App2SFTP()
                sftp.send_SFTP()
        if __name__=='__main__':
                '''
                This runs app2
                '''
                dataJSON = get_connection()
                print(dataJSON)
                hashPayload(dataJSON)
                SFTPSend()
```

app2hash



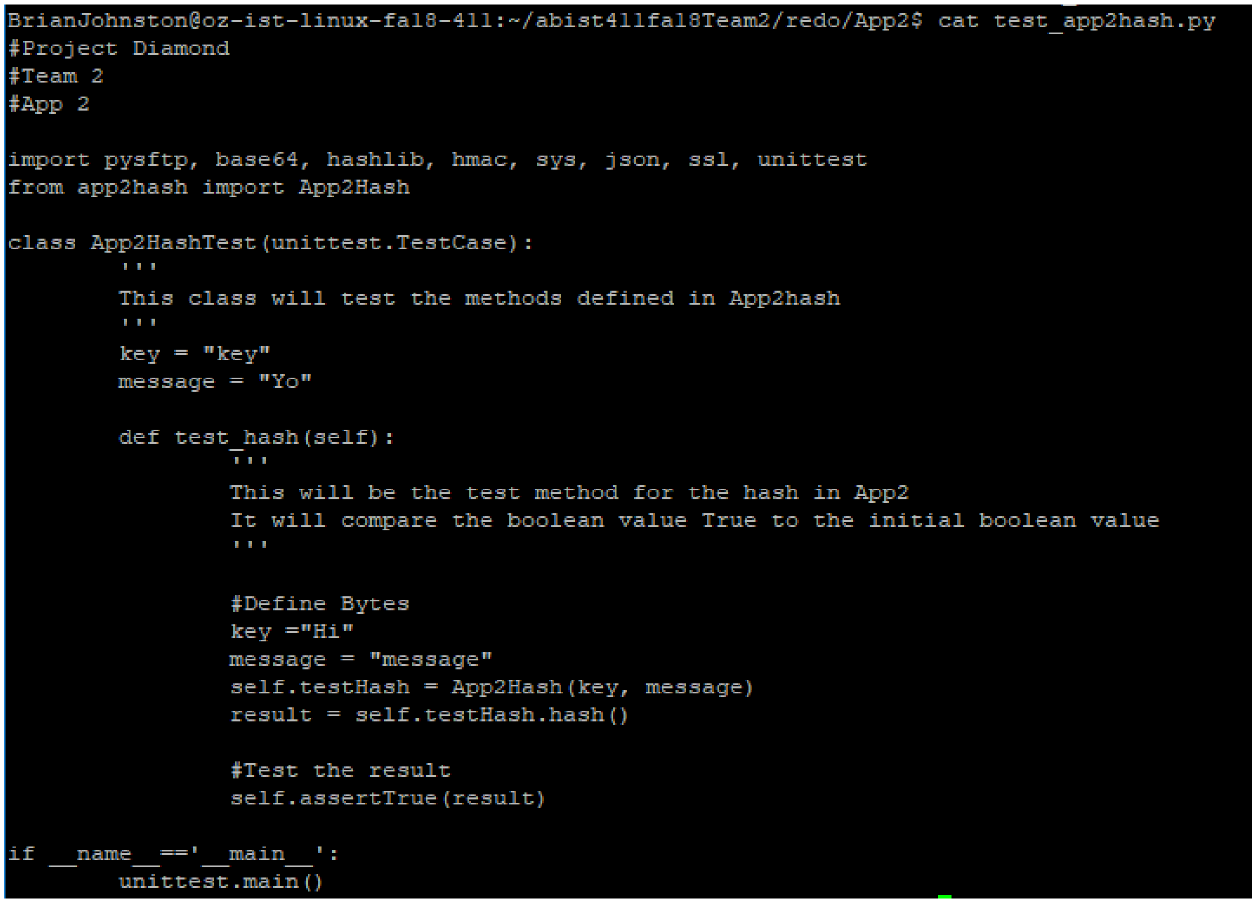
```
BrianJohnston@oz-ist-linux-fa18-411:~/abist411fa18Team2/redo/App2$ cat app2hash.py
#Project: Project Diamond
#Team 2
#App2
import pysftp, base64, hashlib, hmac, sys, socket, ssl, json
sys.path.append("../")
from App5.curlfeed import CurlFeed
class App2Hash:
        '''
        This app hashes the JSON Payload
        '''
        dataJSON = ''
        key = ''
        signature = ''

        def __init__(self,key,dataJSON):
                '''
                Parms :
                Key : the key to hash
                message: message to hmac
                '''
                self.key = key
                self.dataJSON  = dataJSON
                print(type(self.dataJSON))
                print(type(dataJSON))
        def hash(self):
                '''
                Hashes the JSON Payload
                '''
                try:
                        app2key = bytes(self.key,"UTF-8")
                        app2message = bytes(repr(self.dataJSON), "UTF-8")
                        sha256_digester = hmac.new(app2key, app2message, hashlib.sha256)
                        print(sha256_digester)
                        self.sha256_signature = sha256_digester.digest()
                        print("Hashing JSON Payload")
                        print(self.sha256_signature)
                        curlFeed = CurlFeed("App2", "Success", "Successfully hashed the JSON Payload")
                        curlFeed.send()
                        return True
                except:
                        print("Log exception:", sys.exc_info()[0])
                        curlFeed = CurlFeed("App2", "Failure", "Failed to hash the JSON Payload")
                        return False
        def getKey(self):
                return self.key
        def getMessage(self):
                return self.message
        def getSignature(self):
                return self.signature
```

App2Sftp.py



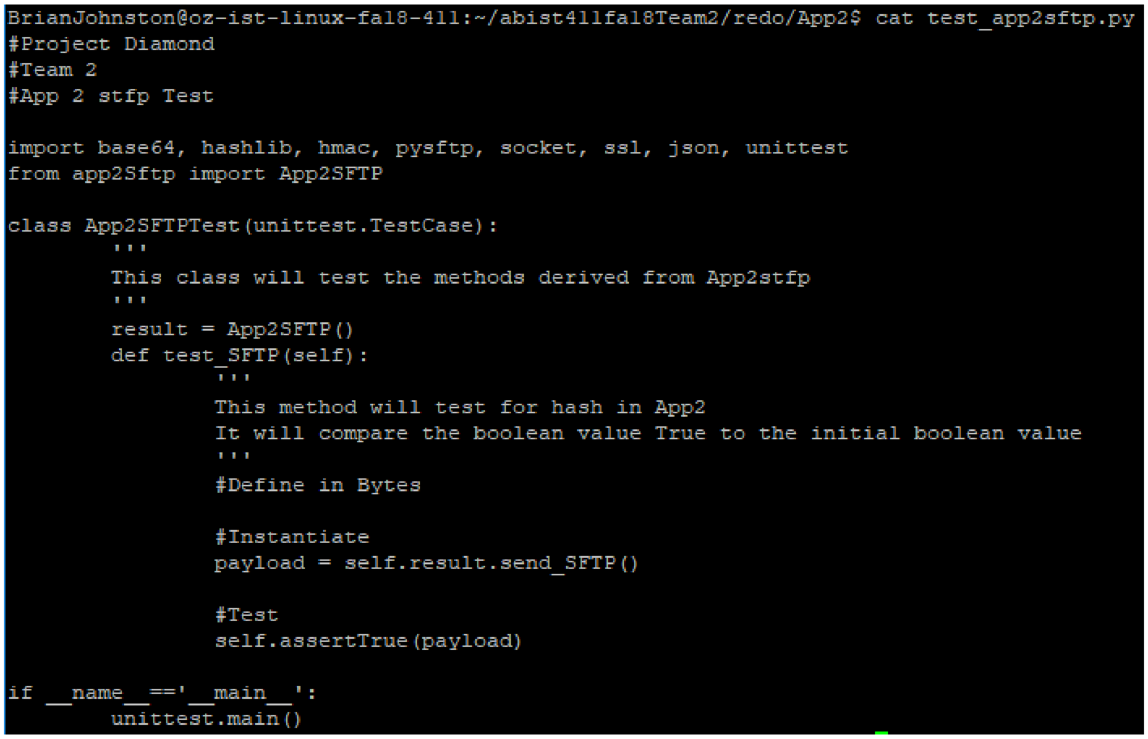
```
BrianJohnston@oz-ist-linux-fa18-411:~/abist411fa18Team2/redo/App2$ cat app2Sftp.py
#Project: Project Diamond
#Team 2
#App2
import sys, pysftp
sys.path.append("../")
from App5.curlfeed import CurlFeed
cnopts = pysftp.CnOpts()
cnopts.hostkeys = None
cinfo = {'cnopts':cnopts, 'host':'oz-ist-linux-fa18-411', 'username':'ftpuser', 'password':'test1234', 'port':103}
class App2SFTP:
        '''
        Description: This sends the payload to App3 using SFTP security
        '''
        def send_SFTP(self):
                '''
                SFTP recieves payload
                '''
                try:
                        with pysftp.Connection(**cinfo) as sftp:
                                print("Connection made")
                                print("Sending the JSON Payload to App3")
                                sftp.put('json.txt')
                                curlFeed = CurlFeed("App2", "Success", "Successfully Sent SFTP Payload to App3")
                                curlFeed.send()
                                return True
                except:
                        print("Log exception:", sys.exc_info()[0])
                        curlFeed = CurlFeed("App2", "Failure", "Failed to send SFTP Payload to App3")
                        curlFeed.send()
```

Test_app2hash.py



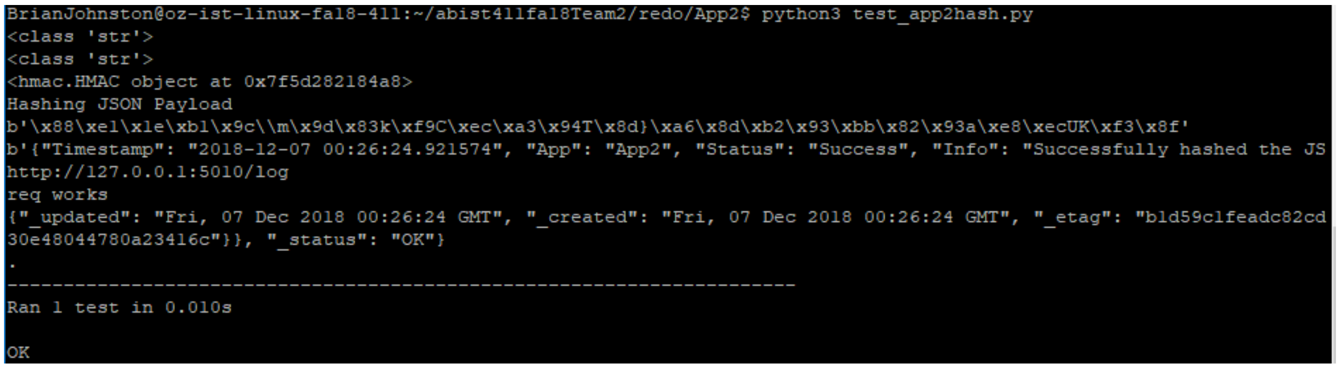
```
BrianJohnston@oz-ist-linux-fa18-411:~/abist411fa18Team2/redo/App2$ cat test_app2hash.py
#Project Diamond
#Team 2
#App 2

import pysftp, base64, hashlib, hmac, sys, json, ssl, unittest
from app2hash import App2Hash

class App2HashTest(unittest.TestCase):
        '''
        This class will test the methods defined in App2hash
        '''
        key = "key"
        message = "Yo"

        def test_hash(self):
                '''
                This will be the test method for the hash in App2
                It will compare the boolean value True to the initial boolean value
                '''

                #Define Bytes
                key ="Hi"
                message = "message"
                self.testHash = App2Hash(key, message)
                result = self.testHash.hash()

                #Test the result
                self.assertTrue(result)

if __name__=='__main__':
        unittest.main()
```

Test_app2sftp.py



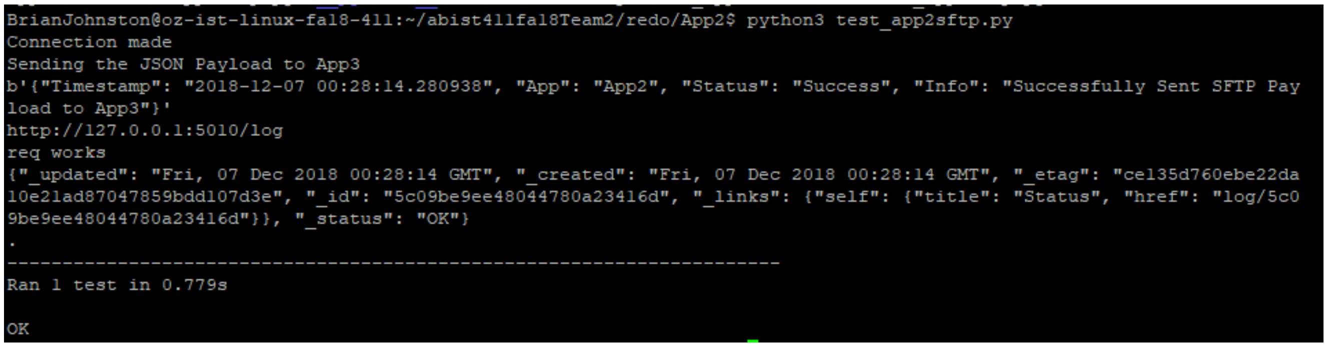
```
BrianJohnston@oz-ist-linux-fa18-411:~/abist411fa18Team2/redo/App2$ cat test_app2sftp.py
#Project Diamond
#Team 2
#App 2 stfp Test

import base64, hashlib, hmac, pysftp, socket, ssl, json, unittest
from app2Sftp import App2SFTP

class App2SFTPTest(unittest.TestCase):
        '''
        This class will test the methods derived from App2stfp
        '''
        result = App2SFTP()
        def test_SFTP(self):
                '''
                This method will test for hash in App2
                It will compare the boolean value True to the initial boolean value
                '''
                #Define in Bytes

                #Instantiate
                payload = self.result.send_SFTP()

                #Test
                self.assertTrue(payload)

if __name__=='__main__':
        unittest.main()
```

Testing app2hash



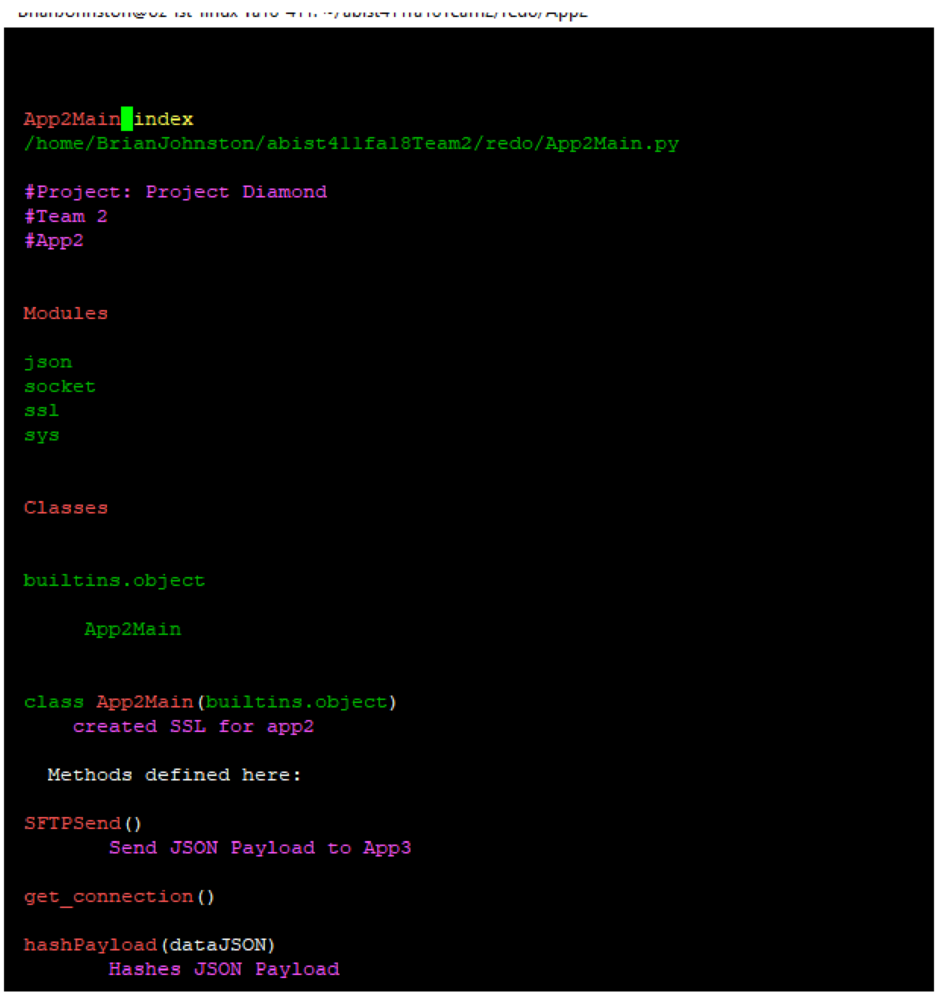
```
BrianJohnston@oz-ist-linux-fa18-411:~/abist411fa18Team2/redo/App2$ python3 test_app2hash.py
<class 'str'>
<class 'str'>
<hmac.HMAC object at 0x7f5d282184a8>
Hashing JSON Payload
b'\x88\xe1\x1e\xb1\x9c\\m\x9d\x83k\xf9C\xec\xa3\x94T\x8d}\xa6\x8d\xb2\x93\xbb\x82\x93a\xe8\xecUK\xf3\x8f'
b'{"Timestamp": "2018-12-07 00:26:24.921574", "App": "App2", "Status": "Success", "Info": "Successfully hashed the JS
http://127.0.0.1:5010/log
req works
{"_updated": "Fri, 07 Dec 2018 00:26:24 GMT", "_created": "Fri, 07 Dec 2018 00:26:24 GMT", "_etag": "b1d59c1feadc82cd
30e48044780a23416c"}}, "_status": "OK"}
.
----------------------------------------------------------------------
Ran 1 test in 0.010s

OK
```

Testing app2sftp



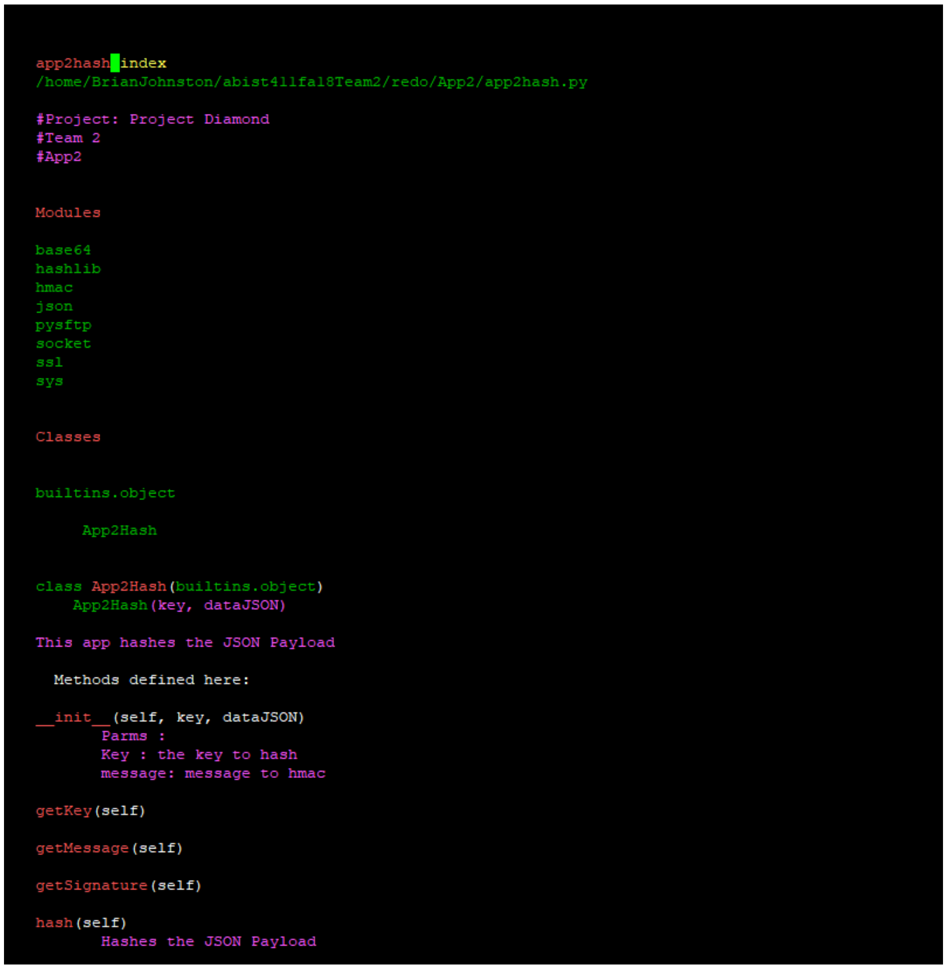
```
BrianJohnston@oz-ist-linux-fa18-411:~/abist411fa18Team2/redo/App2$ python3 test_app2sftp.py
Connection made
Sending the JSON Payload to App3
b'{"Timestamp": "2018-12-07 00:28:14.280938", "App": "App2", "Status": "Success", "Info": "Successfully Sent SFTP Pay
load to App3"}'
http://127.0.0.1:5010/log
req works
{"_updated": "Fri, 07 Dec 2018 00:28:14 GMT", "_created": "Fri, 07 Dec 2018 00:28:14 GMT", "_etag": "ce135d760ebe22da
10e21ad87047859bdd107d3e", "_id": "5c09be9ee48044780a23416d", "_links": {"self": {"title": "Status", "href": "log/5c0
9be9ee48044780a23416d"}}, "_status": "OK"}
.
----------------------------------------------------------------------
Ran 1 test in 0.779s

OK
```

App2Main.html

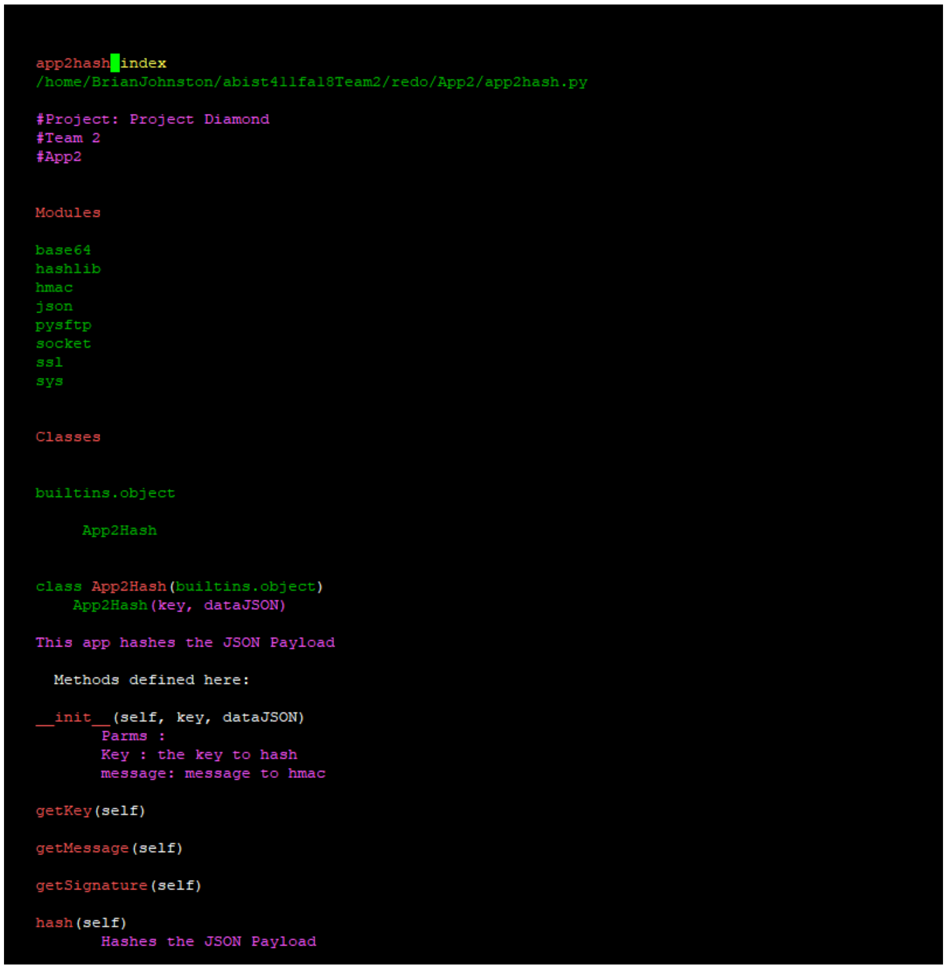


```
App2Main index
/home/BrianJohnston/abist411fa18Team2/redo/App2Main.py

#Project: Project Diamond
#Team 2
#App2


Modules

json
socket
ssl
sys


Classes


builtins.object

     App2Main


class App2Main(builtins.object)
    created SSL for app2

  Methods defined here:

SFTPSend()
       Send JSON Payload to App3

get_connection()

hashPayload(dataJSON)
       Hashes JSON Payload
```

App2hash.html

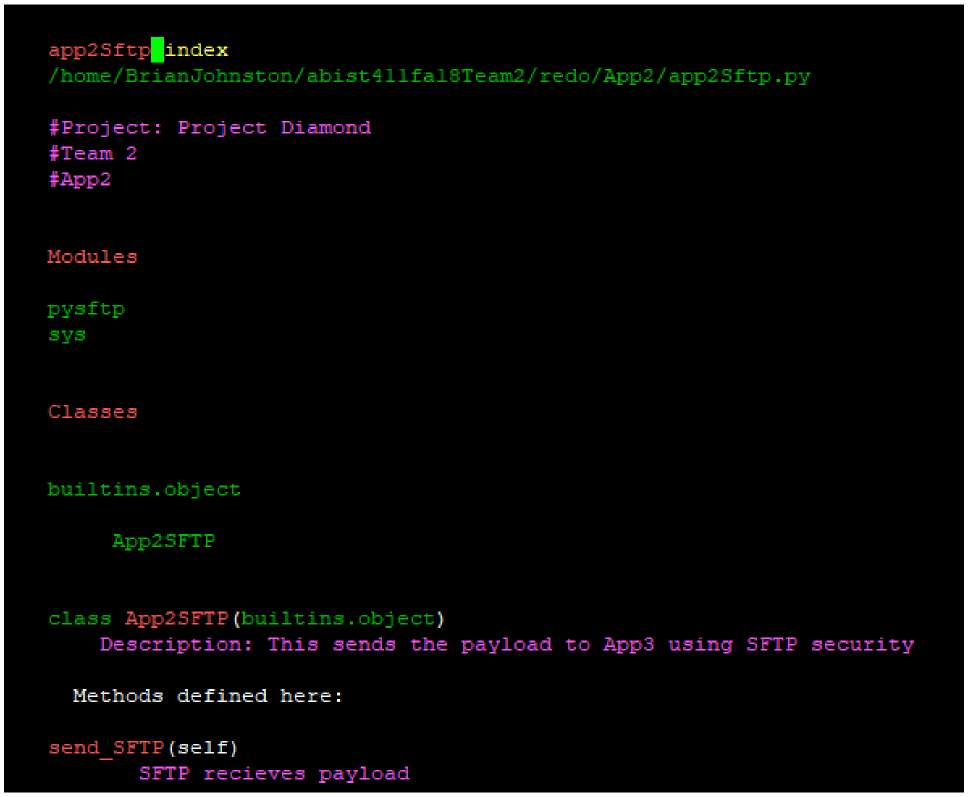


```
app2hash index
/home/BrianJohnston/abist411fa18Team2/redo/App2/app2hash.py

#Project: Project Diamond
#Team 2
#App2


Modules

base64
hashlib
hmac
json
pysftp
socket
ssl
sys


Classes


builtins.object

     App2Hash


class App2Hash(builtins.object)
    App2Hash(key, dataJSON)

This app hashes the JSON Payload

  Methods defined here:

__init__(self, key, dataJSON)
       Parms :
       Key : the key to hash
       message: message to hmac

getKey(self)

getMessage(self)

getSignature(self)

hash(self)
       Hashes the JSON Payload
```



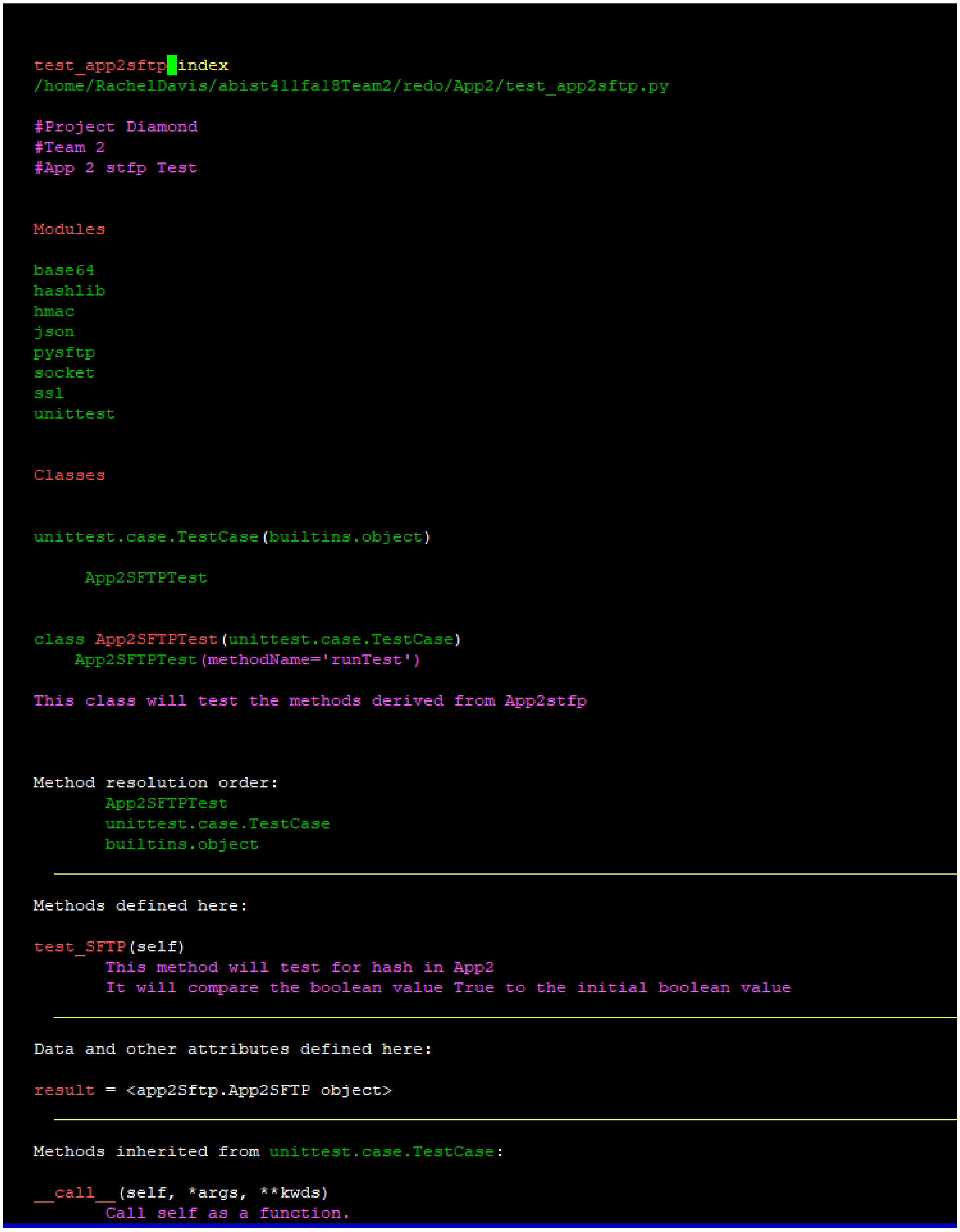
```
app2hash index
/home/BrianJohnston/abist411fa18Team2/redo/App2/app2hash.py

#Project: Project Diamond
#Team 2
#App2


Modules

base64
hashlib
hmac
json
pysftp
socket
ssl
sys


Classes


builtins.object

     App2Hash


class App2Hash(builtins.object)
    App2Hash(key, dataJSON)

This app hashes the JSON Payload

  Methods defined here:

__init__(self, key, dataJSON)
       Parms :
       Key : the key to hash
       message: message to hmac

getKey(self)

getMessage(self)

getSignature(self)

hash(self)
       Hashes the JSON Payload
```

App2Sftp.html



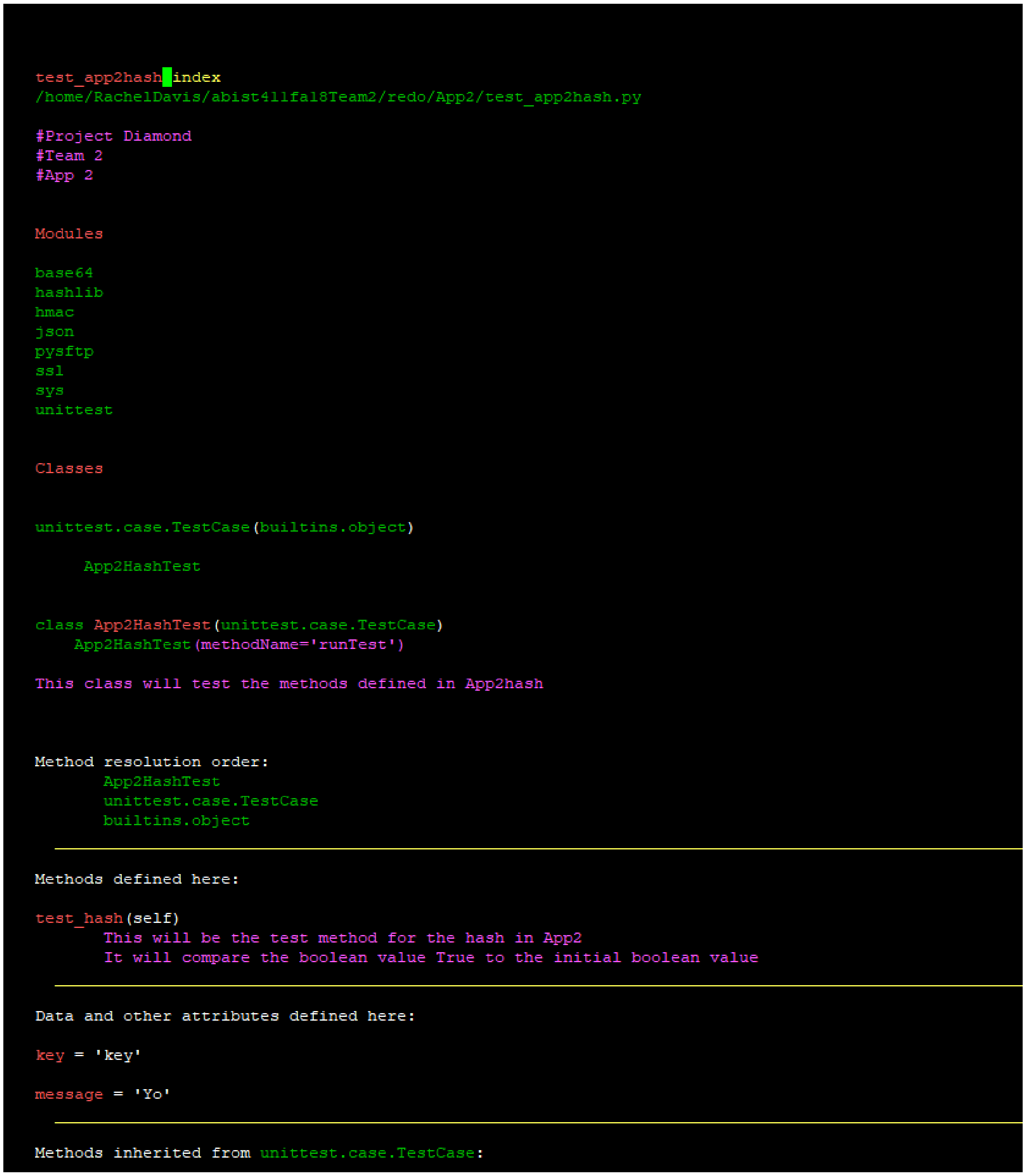
```
app2Sftp index
/home/BrianJohnston/abist411fal8Team2/redo/App2/app2Sftp.py

#Project: Project Diamond
#Team 2
#App2


Modules

pysftp
sys


Classes


builtins.object

     App2SFTP


class App2SFTP(builtins.object)
    Description: This sends the payload to App3 using SFTP security

  Methods defined here:

send_SFTP(self)
       SFTP recieves payload
```

Test_App2sftp.html



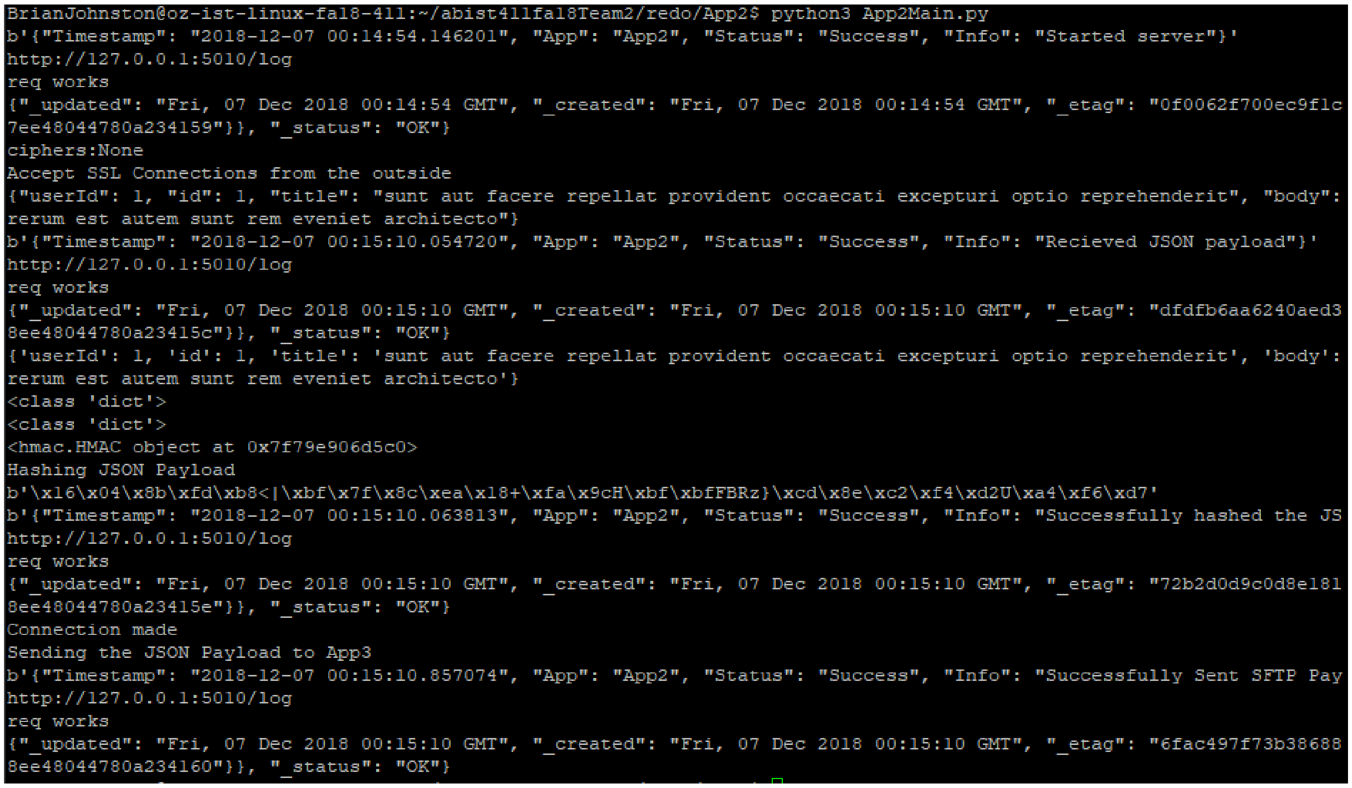
```
test_app2sftp index
/home/RachelDavis/abist411fal8Team2/redo/App2/test_app2sftp.py

#Project Diamond
#Team 2
#App 2 stfp Test


Modules

base64
hashlib
hmac
json
pysftp
socket
ssl
unittest


Classes


unittest.case.TestCase(builtins.object)

     App2SFTPTest


class App2SFTPTest(unittest.case.TestCase)
    App2SFTPTest(methodName='runTest')

This class will test the methods derived from App2stfp



Method resolution order:
       App2SFTPTest
       unittest.case.TestCase
       builtins.object

Methods defined here:

test_SFTP(self)
       This method will test for hash in App2
       It will compare the boolean value True to the initial boolean value

Data and other attributes defined here:

result = <app2Sftp.App2SFTP object>

Methods inherited from unittest.case.TestCase:

__call__(self, *args, **kwds)
       Call self as a function.
```

Test_App2hash.html



```
test_app2hash index
/home/RachelDavis/abist411fa18Team2/redo/App2/test_app2hash.py

#Project Diamond
#Team 2
#App 2


Modules

base64
hashlib
hmac
json
pysftp
ssl
sys
unittest


Classes


unittest.case.TestCase(builtins.object)

     App2HashTest


class App2HashTest(unittest.case.TestCase)
    App2HashTest(methodName='runTest')

This class will test the methods defined in App2hash



Method resolution order:
       App2HashTest
       unittest.case.TestCase
       builtins.object

Methods defined here:

test_hash(self)
       This will be the test method for the hash in App2
       It will compare the boolean value True to the initial boolean value

Data and other attributes defined here:

key = 'key'

message = 'Yo'

Methods inherited from unittest.case.TestCase:
```

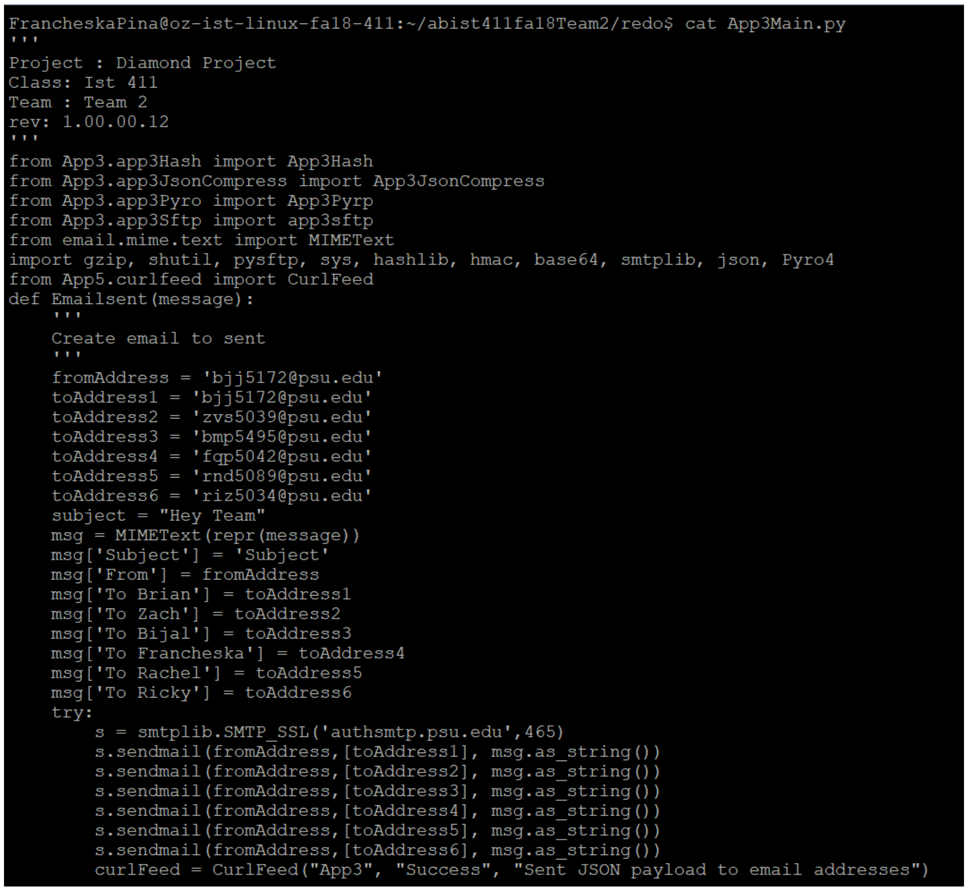
App2 running



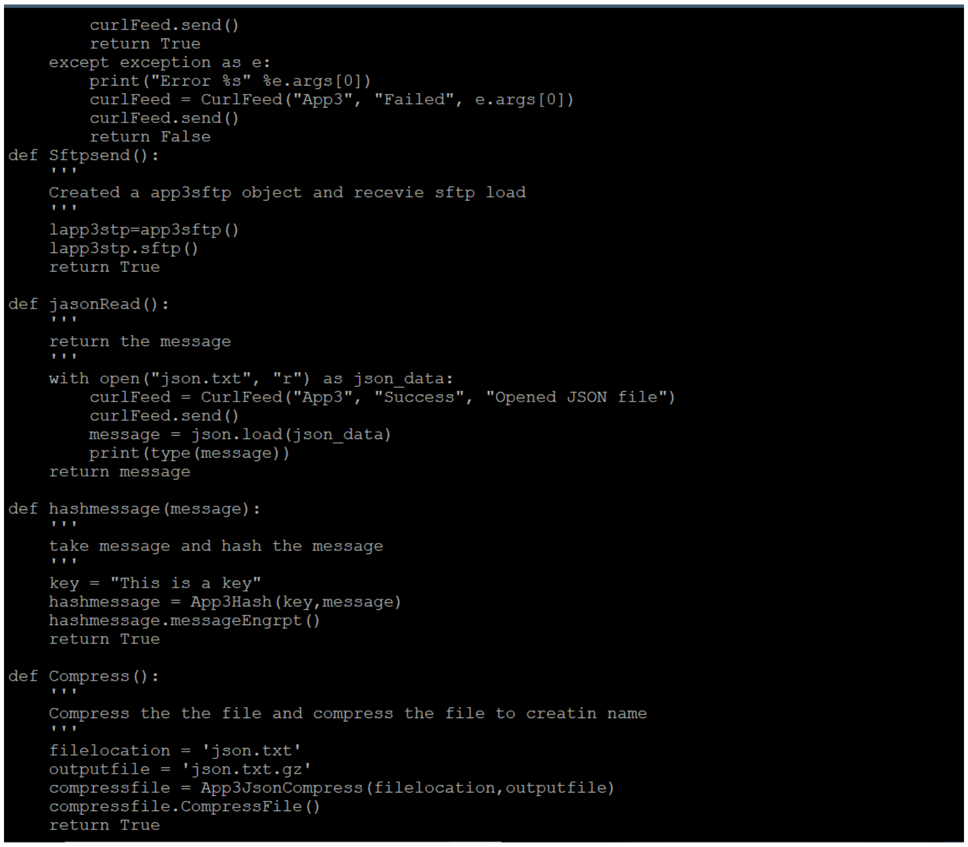
```
BrianJohnston@oz-ist-linux-fa18-411:~/abist411fa18Team2/redo/App2$ python3 App2Main.py
b'{"Timestamp": "2018-12-07 00:14:54.146201", "App": "App2", "Status": "Success", "Info": "Started server"}'
http://127.0.0.1:5010/log
req works
{"_updated": "Fri, 07 Dec 2018 00:14:54 GMT", "_created": "Fri, 07 Dec 2018 00:14:54 GMT", "_etag": "0f0062f700ec9f1c
7ee48044780a234159"}}, "_status": "OK"}
ciphers:None
Accept SSL Connections from the outside
{"userId": 1, "id": 1, "title": "sunt aut facere repellat provident occaecati excepturi optio reprehenderit", "body":
rerum est autem sunt rem eveniet architecto"}
b'{"Timestamp": "2018-12-07 00:15:10.054720", "App": "App2", "Status": "Success", "Info": "Recieved JSON payload"}'
http://127.0.0.1:5010/log
req works
{"_updated": "Fri, 07 Dec 2018 00:15:10 GMT", "_created": "Fri, 07 Dec 2018 00:15:10 GMT", "_etag": "dfdfb6aa6240aed3
8ee48044780a23415c"}}, "_status": "OK"}
{'userId': 1, 'id': 1, 'title': 'sunt aut facere repellat provident occaecati excepturi optio reprehenderit', 'body':
rerum est autem sunt rem eveniet architecto'}
<class 'dict'>
<class 'dict'>
<hmac.HMAC object at 0x7f79e906d5c0>
Hashing JSON Payload
b'\x16\x04\x8b\xfd\xb8<|\xbf\x7f\x8c\xea\x18+\xfa\x9cH\xbf\xbfFBRz}\xcd\x8e\xc2\xf4\xd2U\xa4\xf6\xd7'
b'{"Timestamp": "2018-12-07 00:15:10.063813", "App": "App2", "Status": "Success", "Info": "Successfully hashed the JS
http://127.0.0.1:5010/log
req works
{"_updated": "Fri, 07 Dec 2018 00:15:10 GMT", "_created": "Fri, 07 Dec 2018 00:15:10 GMT", "_etag": "72b2d0d9c0d8e181
8ee48044780a23415e"}}, "_status": "OK"}
Connection made
Sending the JSON Payload to App3
b'{"Timestamp": "2018-12-07 00:15:10.857074", "App": "App2", "Status": "Success", "Info": "Successfully Sent SFTP Pay
http://127.0.0.1:5010/log
req works
{"_updated": "Fri, 07 Dec 2018 00:15:10 GMT", "_created": "Fri, 07 Dec 2018 00:15:10 GMT", "_etag": "6fac497f73b38688
8ee48044780a234160"}}, "_status": "OK"}
```

App 3

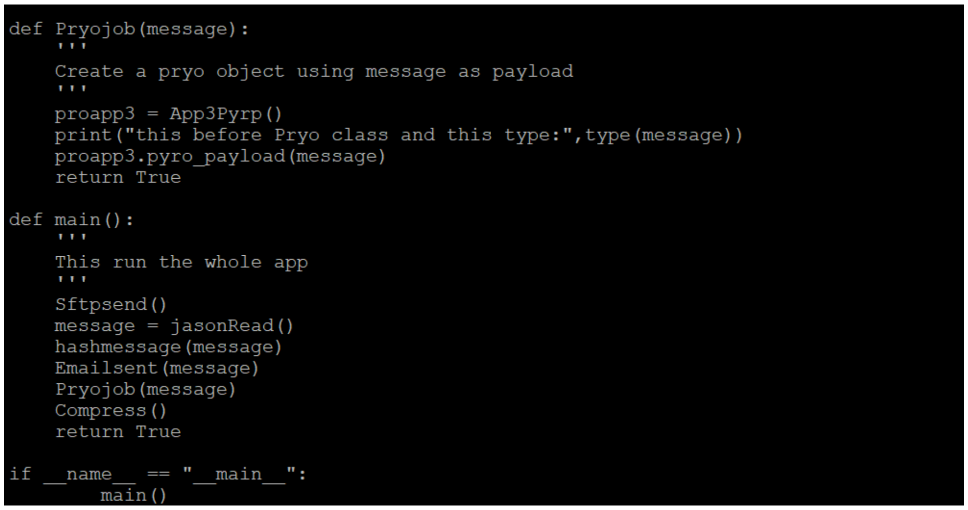
App3 main



```
FrancheskaPina@oz-ist-linux-fa18-411:~/abist411fa18Team2/redo$ cat App3Main.py
'''
Project : Diamond Project
Class: Ist 411
Team : Team 2
rev: 1.00.00.12
'''
from App3.app3Hash import App3Hash
from App3.app3JsonCompress import App3JsonCompress
from App3.app3Pyro import App3Pyrp
from App3.app3Sftp import app3sftp
from email.mime.text import MIMEText
import gzip, shutil, pysftp, sys, hashlib, hmac, base64, smtplib, json, Pyro4
from App5.curlfeed import CurlFeed
def Emailsent(message):
    '''
    Create email to sent
    '''
    fromAddress = 'bjj5172@psu.edu'
    toAddress1 = 'bjj5172@psu.edu'
    toAddress2 = 'zvs5039@psu.edu'
    toAddress3 = 'bmp5495@psu.edu'
    toAddress4 = 'fqp5042@psu.edu'
    toAddress5 = 'rnd5089@psu.edu'
    toAddress6 = 'riz5034@psu.edu'
    subject = "Hey Team"
    msg = MIMEText(repr(message))
    msg['Subject'] = 'Subject'
    msg['From'] = fromAddress
    msg['To Brian'] = toAddress1
    msg['To Zach'] = toAddress2
    msg['To Bijal'] = toAddress3
    msg['To Francheska'] = toAddress4
    msg['To Rachel'] = toAddress5
    msg['To Ricky'] = toAddress6
    try:
        s = smtplib.SMTP_SSL('authsmtp.psu.edu',465)
        s.sendmail(fromAddress,[toAddress1], msg.as_string())
        s.sendmail(fromAddress,[toAddress2], msg.as_string())
        s.sendmail(fromAddress,[toAddress3], msg.as_string())
        s.sendmail(fromAddress,[toAddress4], msg.as_string())
        s.sendmail(fromAddress,[toAddress5], msg.as_string())
        s.sendmail(fromAddress,[toAddress6], msg.as_string())
        curlFeed = CurlFeed("App3", "Success", "Sent JSON payload to email addresses")
```

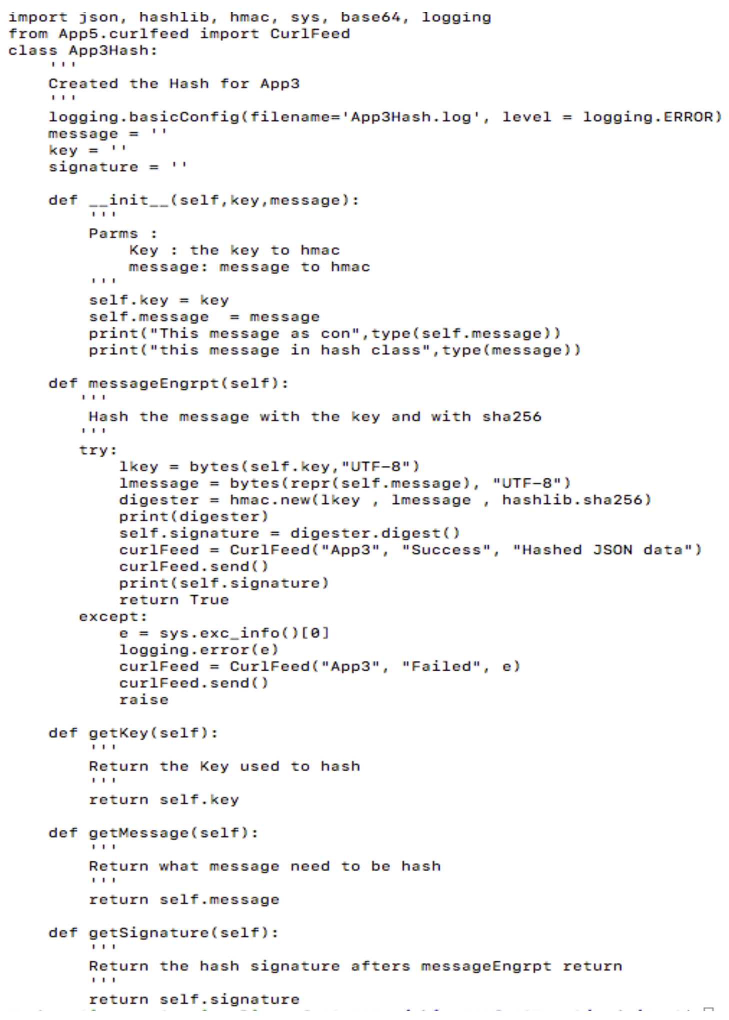


```
        curlFeed.send()
        return True
    except exception as e:
        print("Error %s" %e.args[0])
        curlFeed = CurlFeed("App3", "Failed", e.args[0])
        curlFeed.send()
        return False
def Sftpsend():
    '''
    Created a app3sftp object and recevie sftp load
    '''
    lapp3stp=app3sftp()
    lapp3stp.sftp()
    return True

def jasonRead():
    '''
    return the message
    '''
    with open("json.txt", "r") as json_data:
        curlFeed = CurlFeed("App3", "Success", "Opened JSON file")
        curlFeed.send()
        message = json.load(json_data)
        print(type(message))
    return message

def hashmessage(message):
    '''
    take message and hash the message
    '''
    key = "This is a key"
    hashmessage = App3Hash(key,message)
    hashmessage.messageEngrpt()
    return True

def Compress():
    '''
    Compress the the file and compress the file to creatin name
    '''
    filelocation = 'json.txt'
    outputfile = 'json.txt.gz'
    compressfile = App3JsonCompress(filelocation,outputfile)
    compressfile.CompressFile()
    return True
```



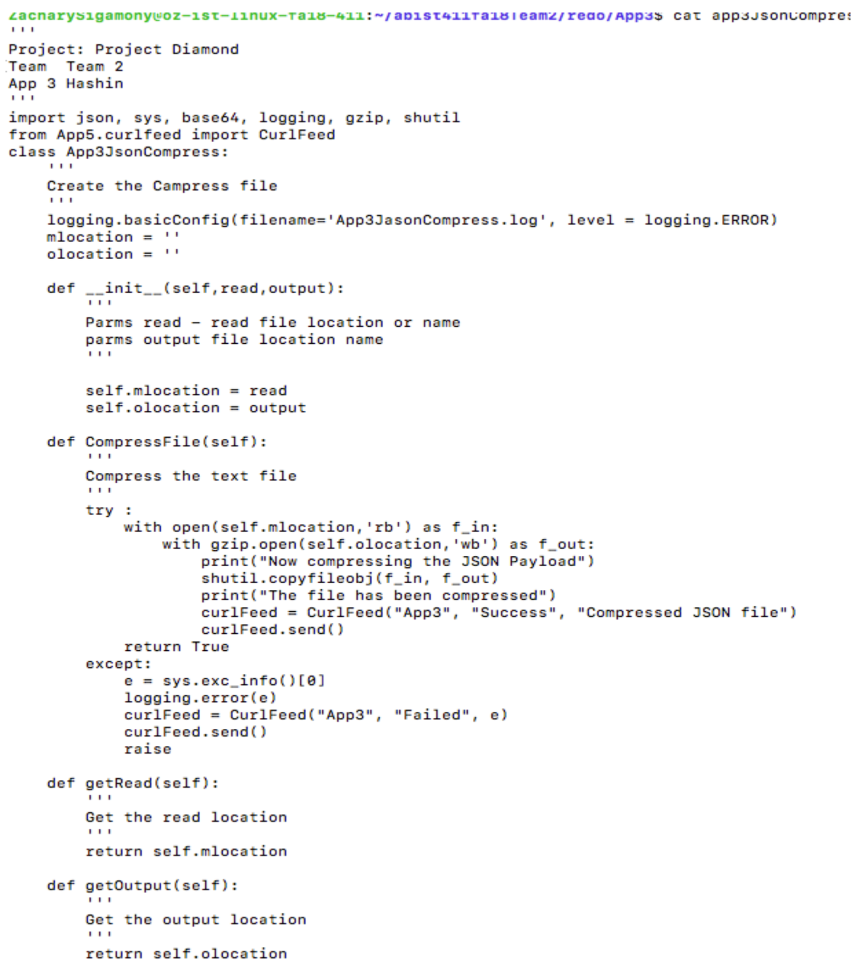
```
def Pryojob(message):
    '''
    Create a pryo object using message as payload
    '''
    proapp3 = App3Pyrp()
    print("this before Pryo class and this type:",type(message))
    proapp3.pyro_payload(message)
    return True

def main():
    '''
    This run the whole app
    '''
    Sftpsend()
    message = jasonRead()
    hashmessage(message)
    Emailsent(message)
    Pryojob(message)
    Compress()
    return True

if __name__ == "__main__":
        main()
```

App3 Hash class



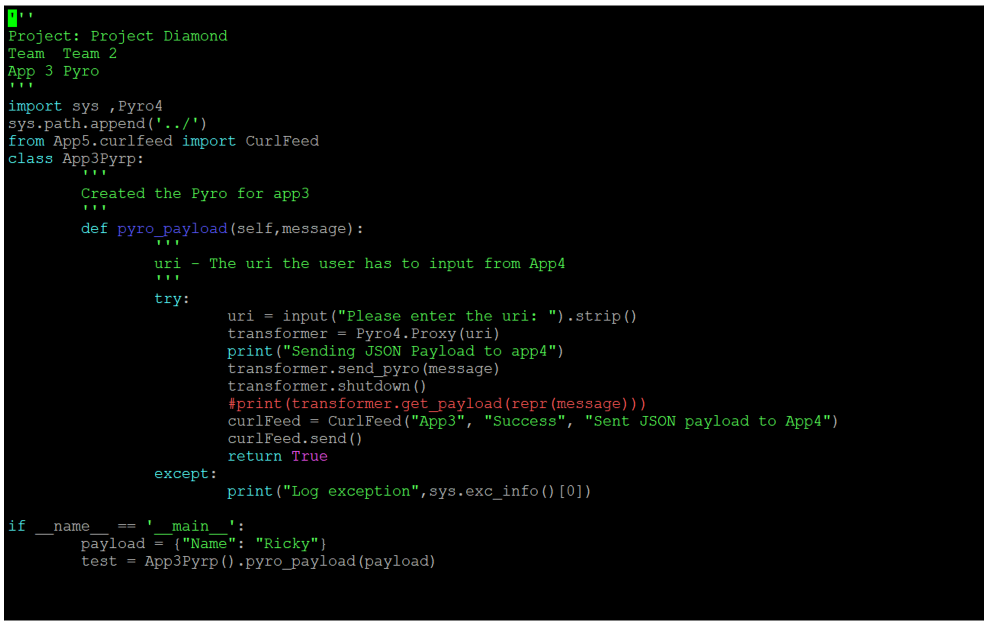
```
import json, hashlib, hmac, sys, base64, logging
from App5.curlfeed import CurlFeed
class App3Hash:
    '''
    Created the Hash for App3
    '''
    logging.basicConfig(filename='App3Hash.log', level = logging.ERROR)
    message = ''
    key = ''
    signature = ''

    def __init__(self,key,message):
        '''
        Parms :
            Key : the key to hmac
            message: message to hmac
        '''
        self.key = key
        self.message  = message
        print("This message as con",type(self.message))
        print("this message in hash class",type(message))

    def messageEngrpt(self):
       '''
        Hash the message with the key and with sha256
       '''
       try:
           lkey = bytes(self.key,"UTF-8")
           lmessage = bytes(repr(self.message), "UTF-8")
           digester = hmac.new(lkey , lmessage , hashlib.sha256)
           print(digester)
           self.signature = digester.digest()
           curlFeed = CurlFeed("App3", "Success", "Hashed JSON data")
           curlFeed.send()
           print(self.signature)
           return True
       except:
           e = sys.exc_info()[0]
           logging.error(e)
           curlFeed = CurlFeed("App3", "Failed", e)
           curlFeed.send()
           raise

    def getKey(self):
        '''
        Return the Key used to hash
        '''
        return self.key

    def getMessage(self):
        '''
        Return what message need to be hash
        '''
        return self.message

    def getSignature(self):
        '''
        Return the hash signature afters messageEngrpt return
        '''
        return self.signature
```

App3 Json class Compress



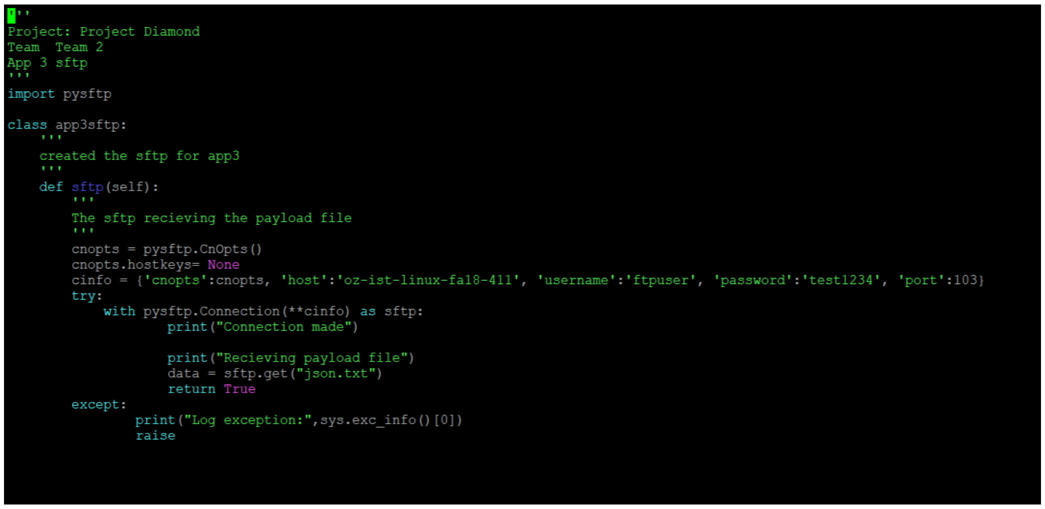
```
zacharysigamony@oz-ist-linux-fa18-411:~/abist411fa18Team2/redo/App3$ cat app3JsonCompre
'''
Project: Project Diamond
Team  Team 2
App 3 Hashin
'''
import json, sys, base64, logging, gzip, shutil
from App5.curlfeed import CurlFeed
class App3JsonCompress:
    '''
    Create the Campress file
    '''
    logging.basicConfig(filename='App3JasonCompress.log', level = logging.ERROR)
    mlocation = ''
    olocation = ''

    def __init__(self,read,output):
        '''
        Parms read - read file location or name
        parms output file location name
        '''

        self.mlocation = read
        self.olocation = output

    def CompressFile(self):
        '''
        Compress the text file
        '''
        try :
            with open(self.mlocation,'rb') as f_in:
                with gzip.open(self.olocation,'wb') as f_out:
                    print("Now compressing the JSON Payload")
                    shutil.copyfileobj(f_in, f_out)
                    print("The file has been compressed")
                    curlFeed = CurlFeed("App3", "Success", "Compressed JSON file")
                    curlFeed.send()
            return True
        except:
            e = sys.exc_info()[0]
            logging.error(e)
            curlFeed = CurlFeed("App3", "Failed", e)
            curlFeed.send()
            raise

    def getRead(self):
        '''
        Get the read location
        '''
        return self.mlocation

    def getOutput(self):
        '''
        Get the output location
        '''
        return self.olocation
```

App3 Pyro class



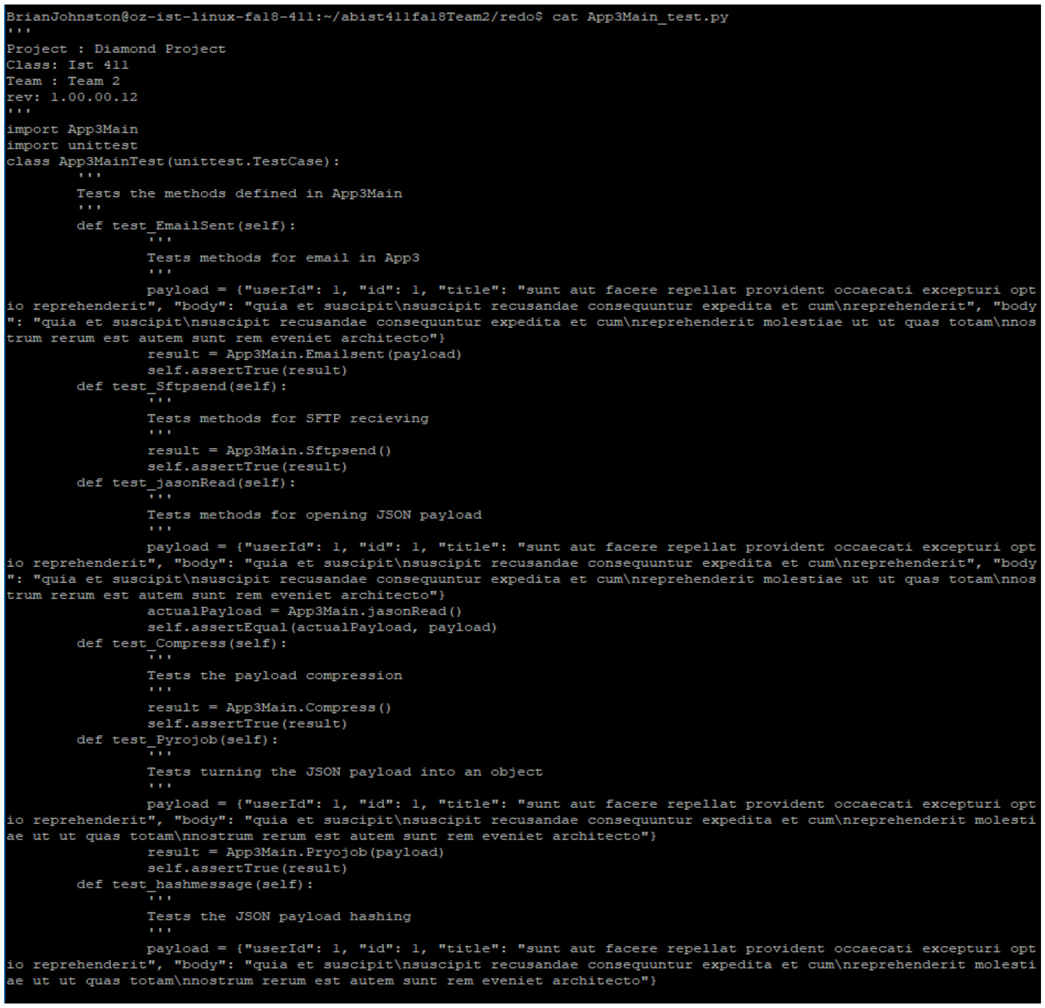
```
'''
Project: Project Diamond
Team  Team 2
App 3 Pyro
'''
import sys ,Pyro4
sys.path.append('../')
from App5.curlfeed import CurlFeed
class App3Pyrp:
        '''
        Created the Pyro for app3
        '''
        def pyro_payload(self,message):
                '''
                uri - The uri the user has to input from App4
                '''
                try:
                        uri = input("Please enter the uri: ").strip()
                        transformer = Pyro4.Proxy(uri)
                        print("Sending JSON Payload to app4")
                        transformer.send_pyro(message)
                        transformer.shutdown()
                        #print(transformer.get_payload(repr(message)))
                        curlFeed = CurlFeed("App3", "Success", "Sent JSON payload to App4")
                        curlFeed.send()
                        return True
                except:
                        print("Log exception",sys.exc_info()[0])

if __name__ == '__main__':
        payload = {"Name": "Ricky"}
        test = App3Pyrp().pyro_payload(payload)
```

App3 Sftp class

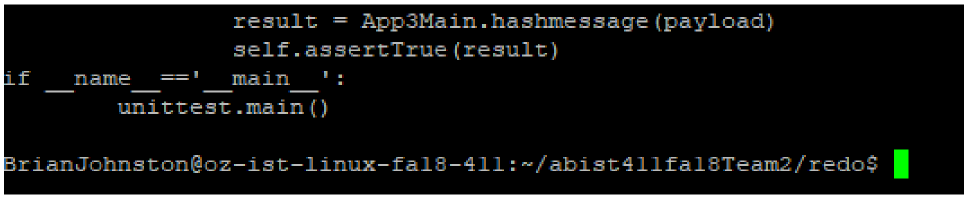


```
'''
Project: Project Diamond
Team  Team 2
App 3 sftp
'''
import pysftp

class app3sftp:
    '''
    created the sftp for app3
    '''
    def sftp(self):
        '''
        The sftp recieving the payload file
        '''
        cnopts = pysftp.CnOpts()
        cnopts.hostkeys= None
        cinfo = {'cnopts':cnopts, 'host':'oz-ist-linux-fa18-411', 'username':'ftpuser', 'password':'test1234', 'port':103}
        try:
            with pysftp.Connection(**cinfo) as sftp:
                    print("Connection made")

                    print("Recieving payload file")
                    data = sftp.get("json.txt")
                    return True
        except:
                print("Log exception:",sys.exc_info()[0])
                raise
```

App3 Main Unit test

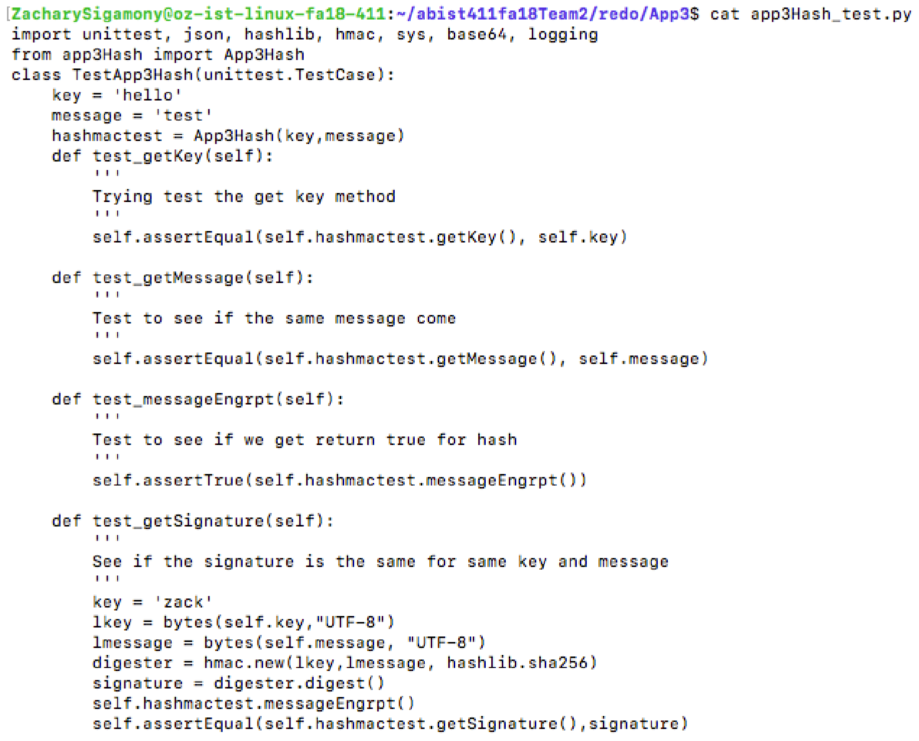


```
BrianJohnston@oz-ist-linux-fa18-411:~/abist411fa18Team2/redo$ cat App3Main_test.py
'''
Project : Diamond Project
Class: Ist 411
Team : Team 2
rev: 1.00.00.12
'''
import App3Main
import unittest
class App3MainTest(unittest.TestCase):
        '''
        Tests the methods defined in App3Main
        '''
        def test_EmailSent(self):
                '''
                Tests methods for email in App3
                '''
                payload = {"userId": 1, "id": 1, "title": "sunt aut facere repellat provident occaecati excepturi opt
io reprehenderit", "body": "quia et suscipit\nsuscipit recusandae consequuntur expedita et cum\nreprehenderit", "body
": "quia et suscipit\nsuscipit recusandae consequuntur expedita et cum\nreprehenderit molestiae ut ut quas totam\nnos
trum rerum est autem sunt rem eveniet architecto"}
                result = App3Main.Emailsent(payload)
                self.assertTrue(result)
        def test_Sftpsend(self):
                '''
                Tests methods for SFTP recieving
                '''
                result = App3Main.Sftpsend()
                self.assertTrue(result)
        def test_jasonRead(self):
                '''
                Tests methods for opening JSON payload
                '''
                payload = {"userId": 1, "id": 1, "title": "sunt aut facere repellat provident occaecati excepturi opt
io reprehenderit", "body": "quia et suscipit\nsuscipit recusandae consequuntur expedita et cum\nreprehenderit", "body
": "quia et suscipit\nsuscipit recusandae consequuntur expedita et cum\nreprehenderit molestiae ut ut quas totam\nnos
trum rerum est autem sunt rem eveniet architecto"}
                actualPayload = App3Main.jasonRead()
                self.assertEqual(actualPayload, payload)
        def test_Compress(self):
                '''
                Tests the payload compression
                '''
                result = App3Main.Compress()
                self.assertTrue(result)
        def test_Pyrojob(self):
                '''
                Tests turning the JSON payload into an object
                '''
                payload = {"userId": 1, "id": 1, "title": "sunt aut facere repellat provident occaecati excepturi opt
io reprehenderit", "body": "quia et suscipit\nsuscipit recusandae consequuntur expedita et cum\nreprehenderit molesti
ae ut ut quas totam\nnostrum rerum est autem sunt rem eveniet architecto"}
                result = App3Main.Pryojob(payload)
                self.assertTrue(result)
        def test_hashmessage(self):
                '''
                Tests the JSON payload hashing
                '''
                payload = {"userId": 1, "id": 1, "title": "sunt aut facere repellat provident occaecati excepturi opt
io reprehenderit", "body": "quia et suscipit\nsuscipit recusandae consequuntur expedita et cum\nreprehenderit molesti
ae ut ut quas totam\nnostrum rerum est autem sunt rem eveniet architecto"}
```



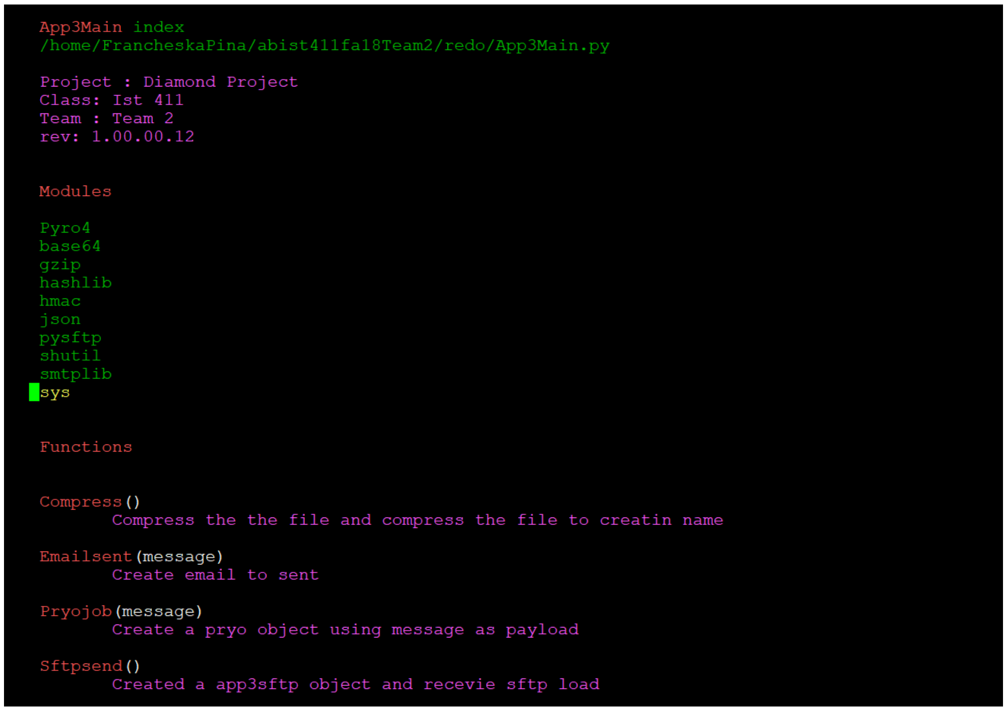
```
                result = App3Main.hashmessage(payload)
                self.assertTrue(result)
if __name__=='__main__':
        unittest.main()

BrianJohnston@oz-ist-linux-fa18-411:~/abist411fa18Team2/redo$
```

App3 Hash class unit test

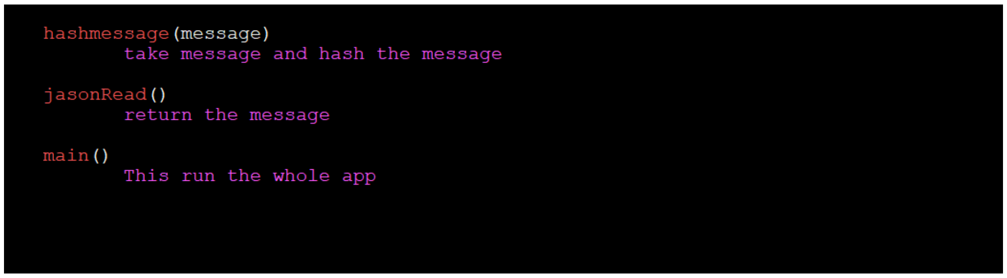


```
[ZacharySigamony@oz-ist-linux-fa18-411:~/abist411fa18Team2/redo/App3$ cat app3Hash_test.py
import unittest, json, hashlib, hmac, sys, base64, logging
from app3Hash import App3Hash
class TestApp3Hash(unittest.TestCase):
    key = 'hello'
    message = 'test'
    hashmactest = App3Hash(key,message)
    def test_getKey(self):
        '''
        Trying test the get key method
        '''
        self.assertEqual(self.hashmactest.getKey(), self.key)

    def test_getMessage(self):
        '''
        Test to see if the same message come
        '''
        self.assertEqual(self.hashmactest.getMessage(), self.message)

    def test_messageEngrpt(self):
        '''
        Test to see if we get return true for hash
        '''
        self.assertTrue(self.hashmactest.messageEngrpt())

    def test_getSignature(self):
        '''
        See if the signature is the same for same key and message
        '''
        key = 'zack'
        lkey = bytes(self.key,"UTF-8")
        lmessage = bytes(self.message, "UTF-8")
        digester = hmac.new(lkey,lmessage, hashlib.sha256)
        signature = digester.digest()
        self.hashmactest.messageEngrpt()
        self.assertEqual(self.hashmactest.getSignature(),signature)
```

App3 main HTML

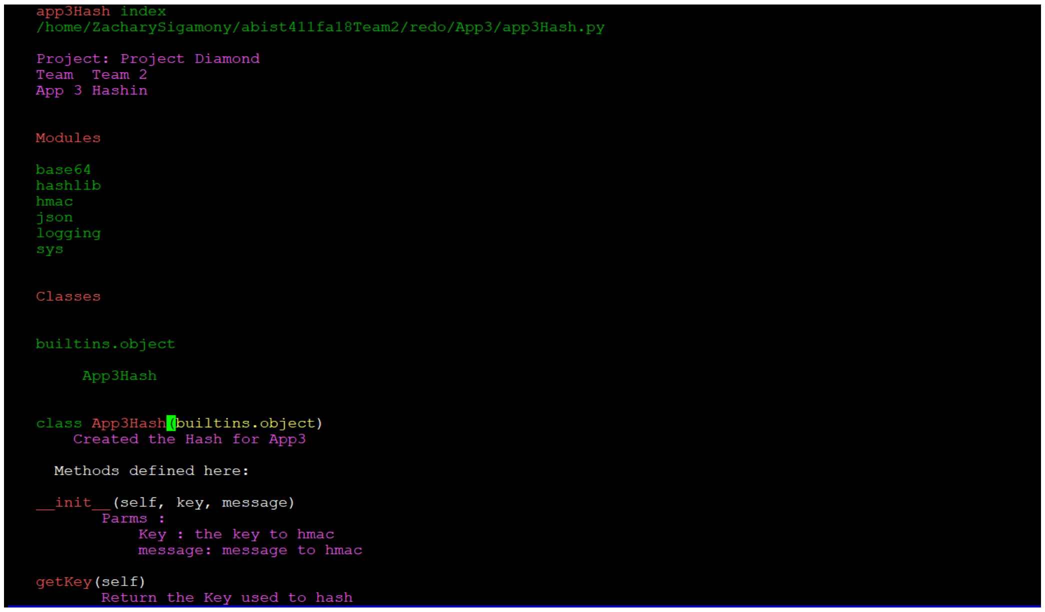


```
App3Main index
/home/FrancheskaPina/abist411fa18Team2/redo/App3Main.py

Project : Diamond Project
Class: Ist 411
Team : Team 2
rev: 1.00.00.12


Modules

Pyro4
base64
gzip
hashlib
hmac
json
pysftp
shutil
smtplib
sys


Functions


Compress()
       Compress the the file and compress the file to creatin name

Emailsent(message)
       Create email to sent

Pryojob(message)
       Create a pryo object using message as payload

Sftpsend()
       Created a app3sftp object and recevie sftp load
```

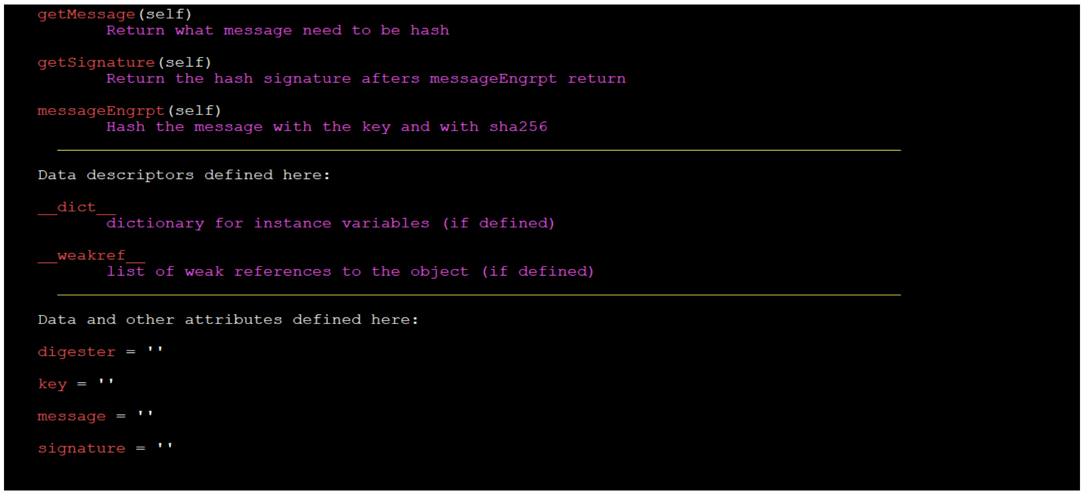


```
hashmessage(message)
       take message and hash the message

jasonRead()
       return the message

main()
       This run the whole app
```

App3 Hash HTML

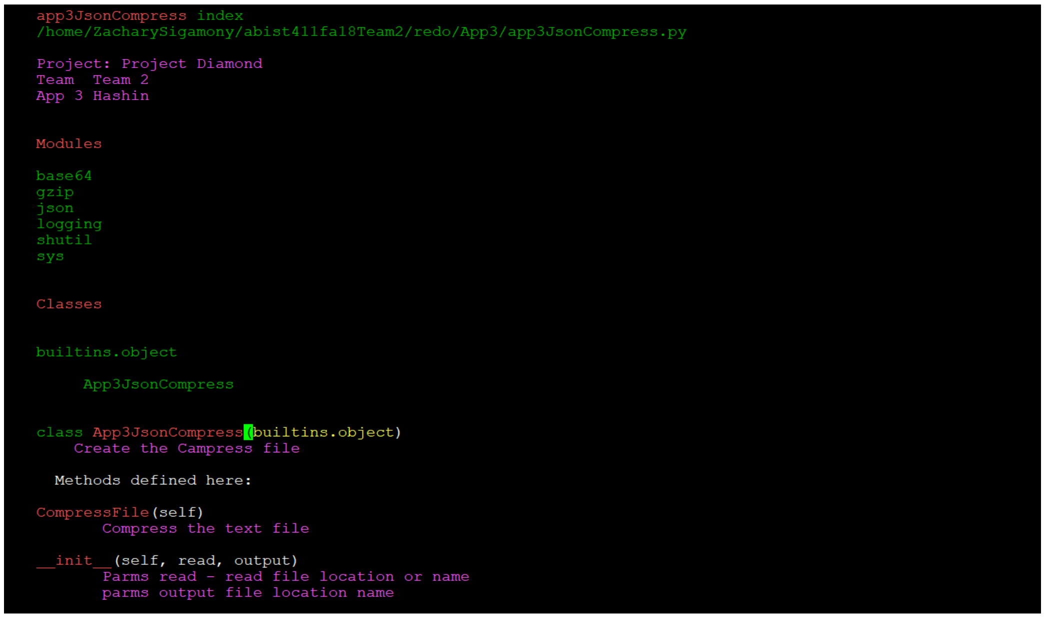


```
app3Hash index
/home/ZacharySigamony/abist411fa18Team2/redo/App3/app3Hash.py

Project: Project Diamond
Team  Team 2
App 3 Hashin


Modules

base64
hashlib
hmac
json
logging
sys


Classes


builtins.object

     App3Hash


class App3Hash(builtins.object)
    Created the Hash for App3

  Methods defined here:

__init__(self, key, message)
       Parms :
           Key : the key to hmac
           message: message to hmac

getKey(self)
       Return the Key used to hash
```

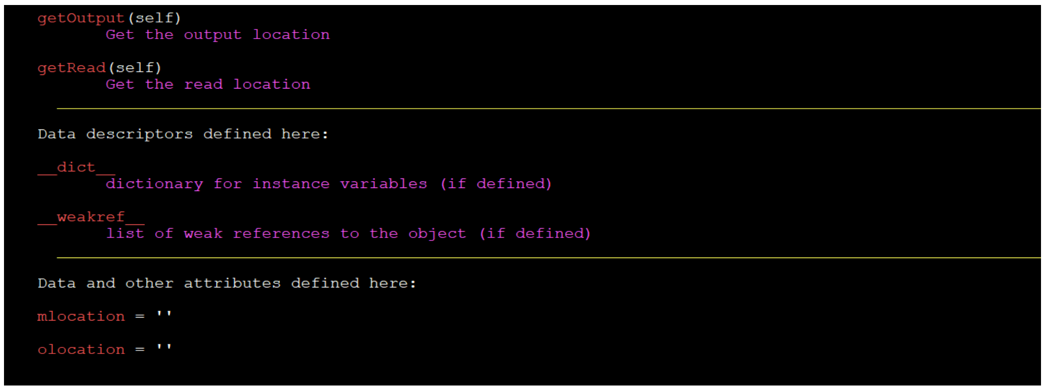


```
getMessage(self)
       Return what message need to be hash

getSignature(self)
       Return the hash signature afters messageEngrpt return

messageEngrpt(self)
       Hash the message with the key and with sha256

Data descriptors defined here:

__dict__
       dictionary for instance variables (if defined)

__weakref__
       list of weak references to the object (if defined)

Data and other attributes defined here:

digester = ''

key = ''

message = ''

signature = ''
```

App3 Json Compress HTML

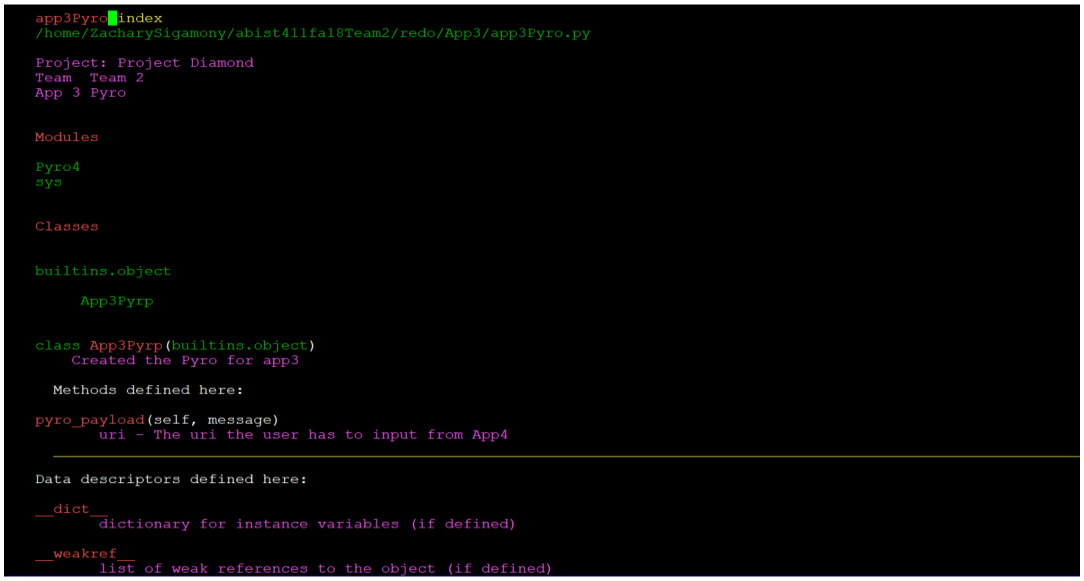


```
app3JsonCompress index
/home/ZacharySigamony/abist411fa18Team2/redo/App3/app3JsonCompress.py

Project: Project Diamond
Team  Team 2
App 3 Hashin


Modules

base64
gzip
json
logging
shutil
sys


Classes


builtins.object

     App3JsonCompress


class App3JsonCompress(builtins.object)
    Create the Campress file

  Methods defined here:

CompressFile(self)
       Compress the text file

__init__(self, read, output)
       Parms read - read file location or name
       parms output file location name
```



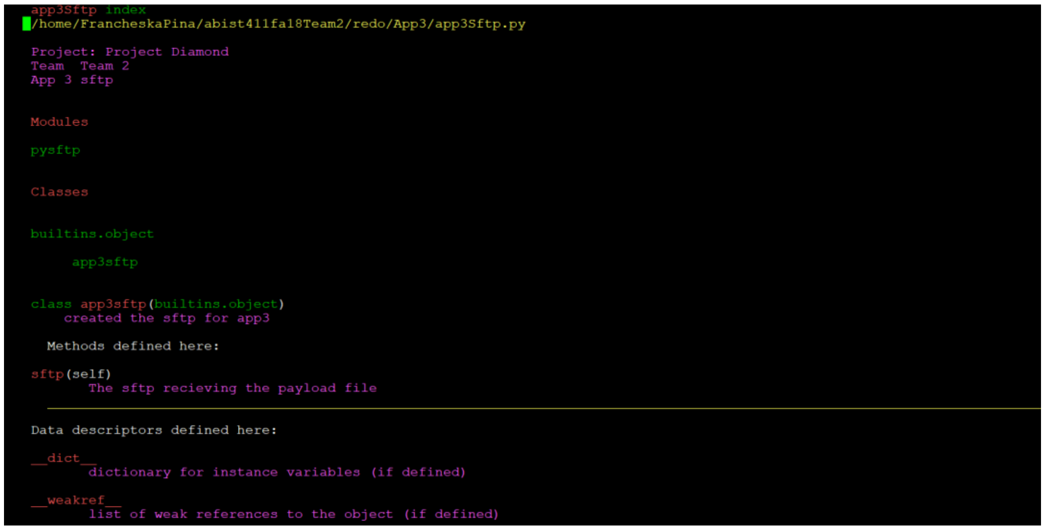
```
getOutput(self)
       Get the output location

getRead(self)
       Get the read location

Data descriptors defined here:

__dict__
       dictionary for instance variables (if defined)

__weakref__
       list of weak references to the object (if defined)

Data and other attributes defined here:

mlocation = ''

olocation = ''
```

App3 Pyro HTML



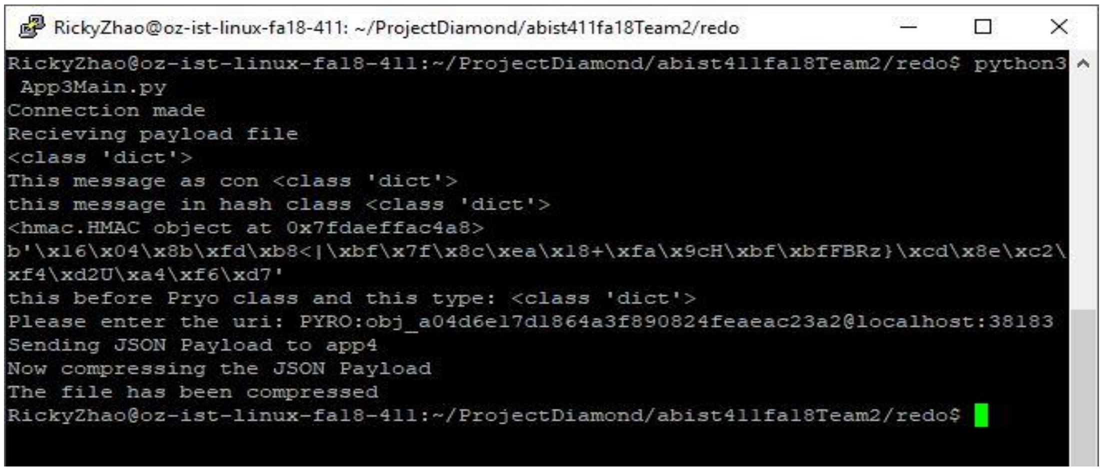
```
app3Pyro index
/home/ZacharySigamony/abist411fa18Team2/redo/App3/app3Pyro.py

Project: Project Diamond
Team  Team 2
App 3 Pyro


Modules

Pyro4
sys


Classes


builtins.object

     App3Pyrp


class App3Pyrp(builtins.object)
    Created the Pyro for app3

  Methods defined here:

pyro_payload(self, message)
       uri - The uri the user has to input from App4

Data descriptors defined here:

__dict__
       dictionary for instance variables (if defined)

__weakref__
       list of weak references to the object (if defined)
```

App3 Sftp HTML



```
app3Sftp index
/home/FrancheskaPina/abist411fa18Team2/redo/App3/app3Sftp.py

Project: Project Diamond
Team  Team 2
App 3 sftp


Modules

pysftp


Classes


builtins.object

     app3sftp


class app3sftp(builtins.object)
    created the sftp for app3

  Methods defined here:

sftp(self)
       The sftp recieving the payload file

Data descriptors defined here:

__dict__
       dictionary for instance variables (if defined)

__weakref__
       list of weak references to the object (if defined)
```

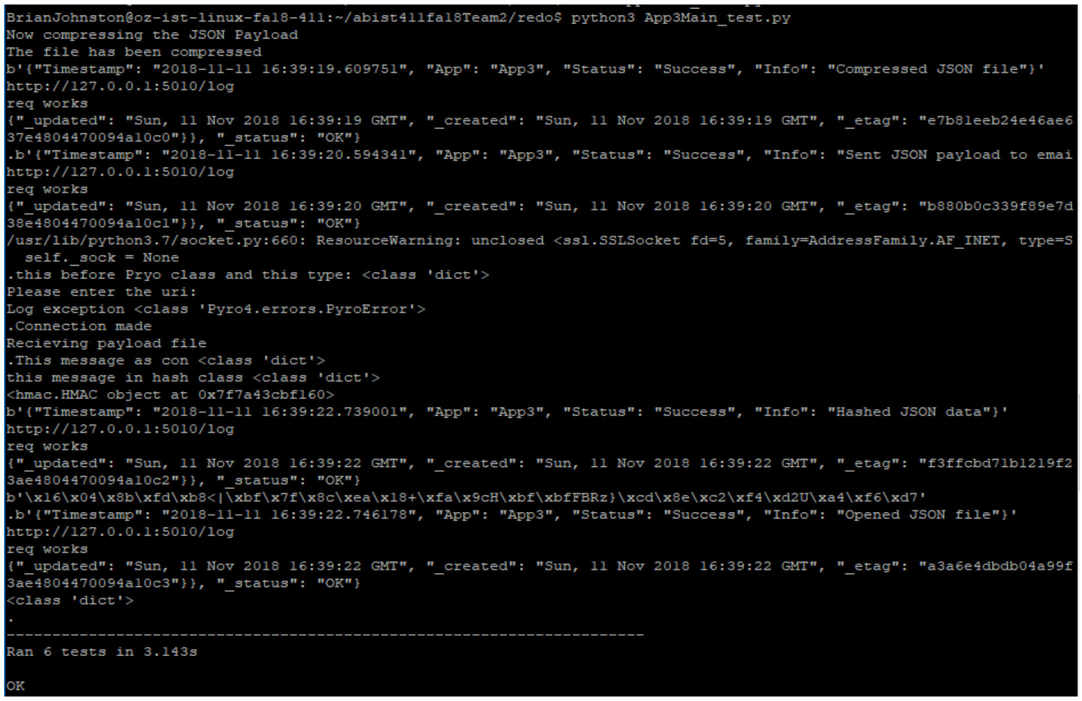
App3 main running



RickyZhao@oz-ist-linux-fa18-411: ~/ProjectDiamond/abist411fa18Team2/redo

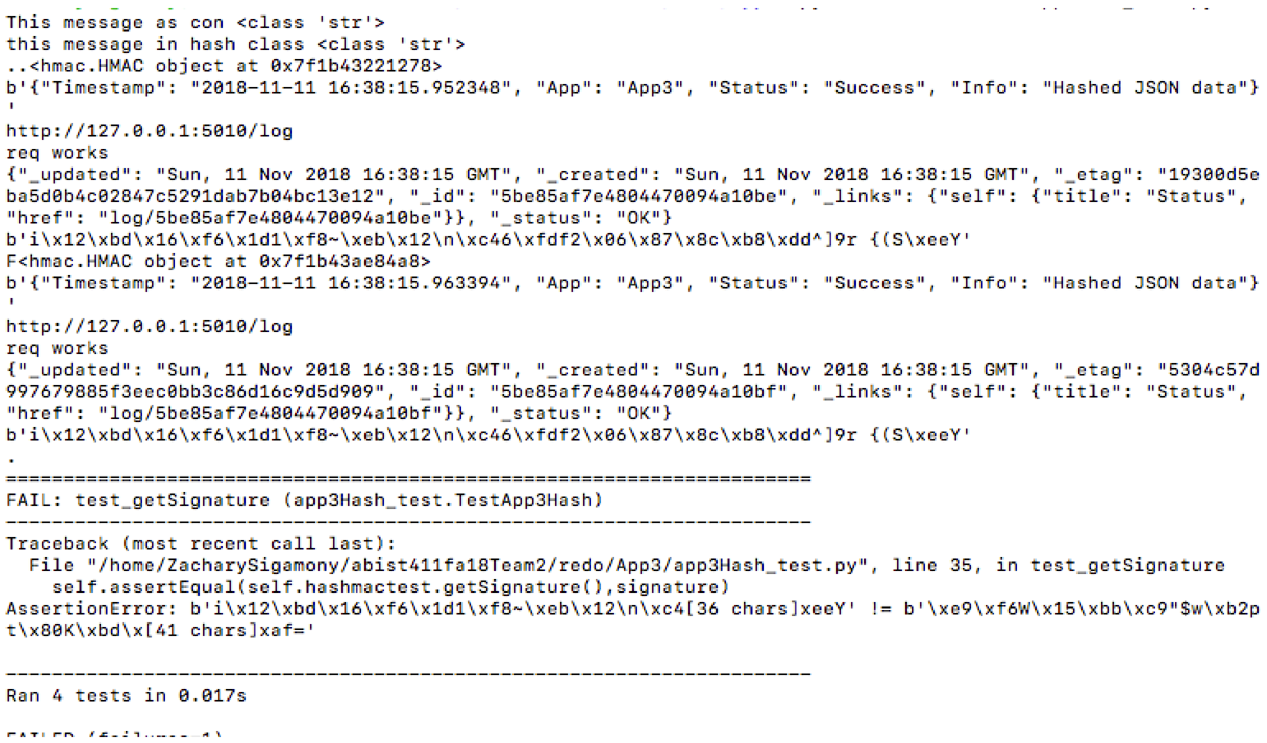
```
RickyZhao@oz-ist-linux-fa18-411:~/ProjectDiamond/abist411fa18Team2/redo$ python3
 App3Main.py
Connection made
Recieving payload file
<class 'dict'>
This message as con <class 'dict'>
this message in hash class <class 'dict'>
<hmac.HMAC object at 0x7fdaeffac4a8>
b'\x16\x04\x8b\xfd\xb8<|\xbf\x7f\x8c\xea\x18+\xfa\x9cH\xbf\xbfFBRz}\xcd\x8e\xc2\
xf4\xd2U\xa4\xf6\xd7'
this before Pryo class and this type: <class 'dict'>
Please enter the uri: PYRO:obj_a04d6e17d1864a3f890824feaeac23a2@localhost:38183
Sending JSON Payload to app4
Now compressing the JSON Payload
The file has been compressed
RickyZhao@oz-ist-linux-fa18-411:~/ProjectDiamond/abist411fa18Team2/redo$
```

App3 main unit test running



```
BrianJohnston@oz-ist-linux-fa18-411:~/abist411fa18Team2/redo$ python3 App3Main_test.py
Now compressing the JSON Payload
The file has been compressed
b'{"Timestamp": "2018-11-11 16:39:19.609751", "App": "App3", "Status": "Success", "Info": "Compressed JSON file"}'
http://127.0.0.1:5010/log
req works
{"_updated": "Sun, 11 Nov 2018 16:39:19 GMT", "_created": "Sun, 11 Nov 2018 16:39:19 GMT", "_etag": "e7b81eeb24e46ae6
37e4804470094a10c0"}}, "_status": "OK"}
.b'{"Timestamp": "2018-11-11 16:39:20.594341", "App": "App3", "Status": "Success", "Info": "Sent JSON payload to emai
http://127.0.0.1:5010/log
req works
{"_updated": "Sun, 11 Nov 2018 16:39:20 GMT", "_created": "Sun, 11 Nov 2018 16:39:20 GMT", "_etag": "b880b0c339f89e7d
38e4804470094a10c1"}}, "_status": "OK"}
/usr/lib/python3.7/socket.py:660: ResourceWarning: unclosed <ssl.SSLSocket fd=5, family=AddressFamily.AF_INET, type=S
  self._sock = None
.this before Pryo class and this type: <class 'dict'>
Please enter the uri:
Log exception <class 'Pyro4.errors.PyroError'>
.Connection made
Recieving payload file
.This message as con <class 'dict'>
this message in hash class <class 'dict'>
<hmac.HMAC object at 0x7f7a43cbf160>
b'{"Timestamp": "2018-11-11 16:39:22.739001", "App": "App3", "Status": "Success", "Info": "Hashed JSON data"}'
http://127.0.0.1:5010/log
req works
{"_updated": "Sun, 11 Nov 2018 16:39:22 GMT", "_created": "Sun, 11 Nov 2018 16:39:22 GMT", "_etag": "f3ffcbd71b1219f2
3ae4804470094a10c2"}}, "_status": "OK"}
b'\x16\x04\x8b\xfd\xb8<|\xbf\x7f\x8c\xea\x18+\xfa\x9cH\xbf\xbfFBRz}\xcd\x8e\xc2\xf4\xd2U\xa4\xf6\xd7'
.b'{"Timestamp": "2018-11-11 16:39:22.746178", "App": "App3", "Status": "Success", "Info": "Opened JSON file"}'
http://127.0.0.1:5010/log
req works
{"_updated": "Sun, 11 Nov 2018 16:39:22 GMT", "_created": "Sun, 11 Nov 2018 16:39:22 GMT", "_etag": "a3a6e4dbdb04a99f
3ae4804470094a10c3"}}, "_status": "OK"}
<class 'dict'>
.
----------------------------------------------------------------------
Ran 6 tests in 3.143s

OK
```

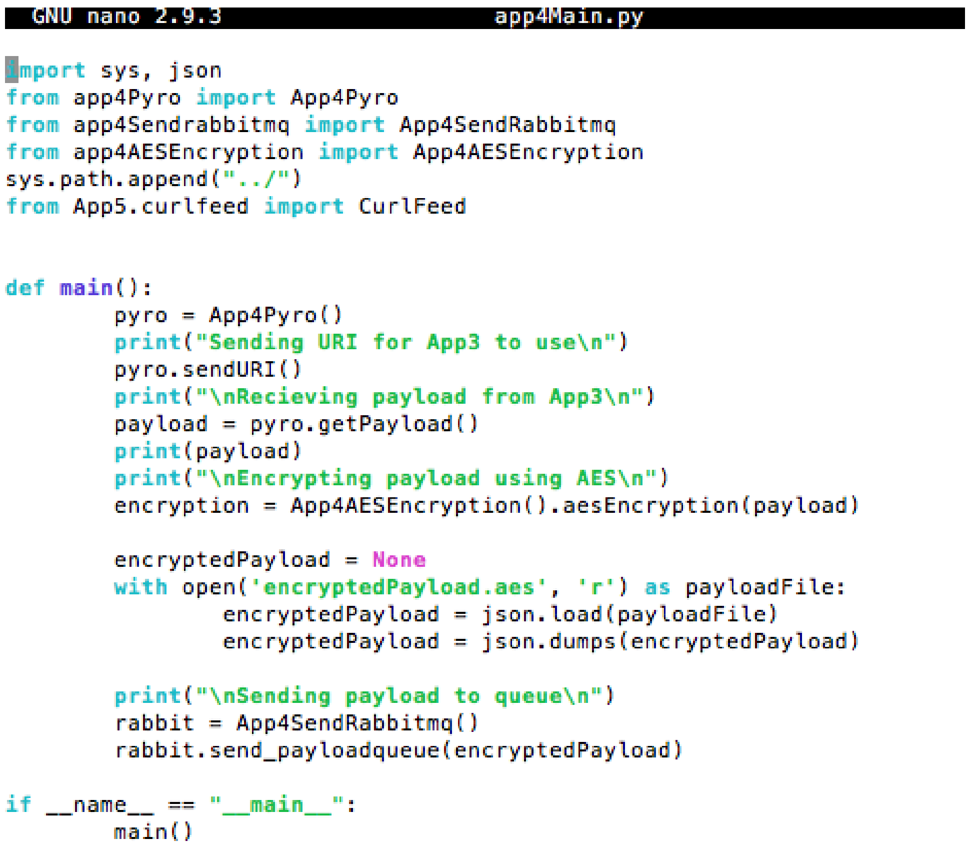
App3 Hash unit test running



```
This message as con <class 'str'>
this message in hash class <class 'str'>
..<hmac.HMAC object at 0x7f1b43221278>
b'{"Timestamp": "2018-11-11 16:38:15.952348", "App": "App3", "Status": "Success", "Info": "Hashed JSON data"}
'
http://127.0.0.1:5010/log
req works
{"_updated": "Sun, 11 Nov 2018 16:38:15 GMT", "_created": "Sun, 11 Nov 2018 16:38:15 GMT", "_etag": "19300d5e
ba5d0b4c02847c5291dab7b04bc13e12", "_id": "5be85af7e4804470094a10be", "_links": {"self": {"title": "Status",
"href": "log/5be85af7e4804470094a10be"}}, "_status": "OK"}
b'i\x12\xbd\x16\xf6\x1d1\xf8~\xeb\x12\n\xc46\xfdf2\x06\x87\x8c\xb8\xdd^]9r {(S\xeeY'
F<hmac.HMAC object at 0x7f1b43ae84a8>
b'{"Timestamp": "2018-11-11 16:38:15.963394", "App": "App3", "Status": "Success", "Info": "Hashed JSON data"}
'
http://127.0.0.1:5010/log
req works
{"_updated": "Sun, 11 Nov 2018 16:38:15 GMT", "_created": "Sun, 11 Nov 2018 16:38:15 GMT", "_etag": "5304c57d
997679885f3eec0bb3c86d16c9d5d909", "_id": "5be85af7e4804470094a10bf", "_links": {"self": {"title": "Status",
"href": "log/5be85af7e4804470094a10bf"}}, "_status": "OK"}
b'i\x12\xbd\x16\xf6\x1d1\xf8~\xeb\x12\n\xc46\xfdf2\x06\x87\x8c\xb8\xdd^]9r {(S\xeeY'
.
======================================================================
FAIL: test_getSignature (app3Hash_test.TestApp3Hash)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/home/ZacharySigamony/abist411fa18Team2/redo/App3/app3Hash_test.py", line 35, in test_getSignature
    self.assertEqual(self.hashmactest.getSignature(),signature)
AssertionError: b'i\x12\xbd\x16\xf6\x1d1\xf8~\xeb\x12\n\xc4[36 chars]xeeY' != b'\xe9\xf6W\x15\xbb\xc9"$w\xb2p
t\x80K\xbd\x[41 chars]xaf='

----------------------------------------------------------------------
Ran 4 tests in 0.017s

FAILED (failures=1)
```

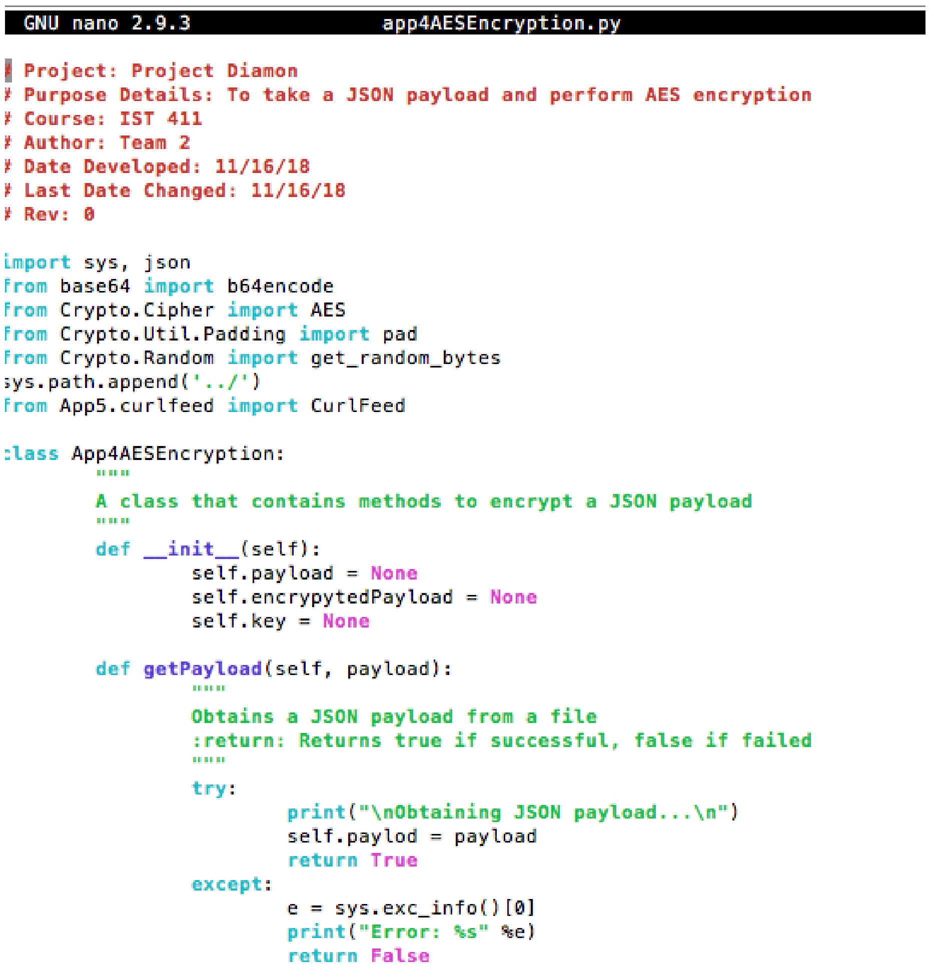
App4

App4 main

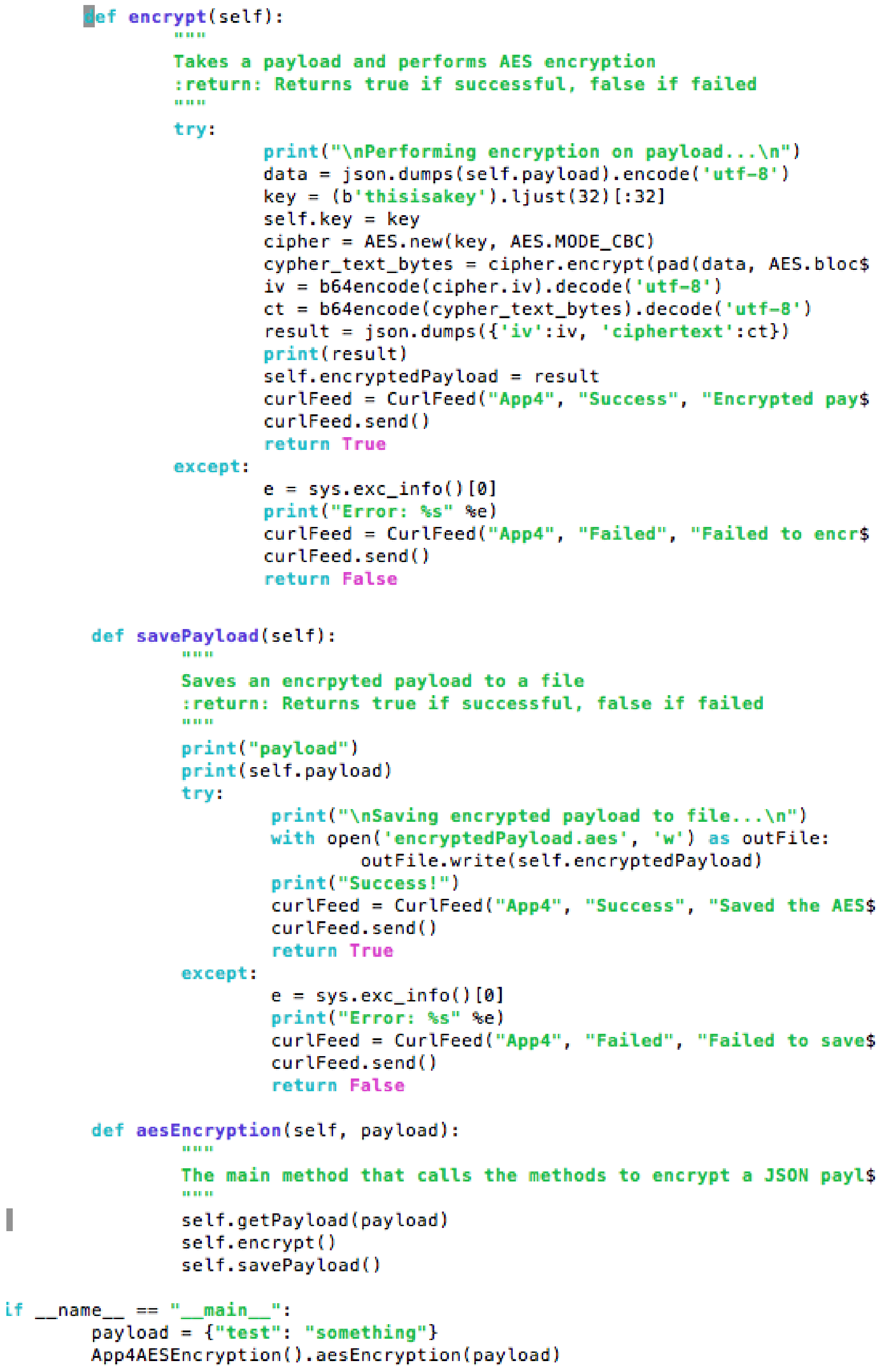


```
  GNU nano 2.9.3                              app4Main.py

import sys, json
from app4Pyro import App4Pyro
from app4Sendrabbitmq import App4SendRabbitmq
from app4AESEncryption import App4AESEncryption
sys.path.append("../")
from App5.curlfeed import CurlFeed


def main():
        pyro = App4Pyro()
        print("Sending URI for App3 to use\n")
        pyro.sendURI()
        print("\nRecieving payload from App3\n")
        payload = pyro.getPayload()
        print(payload)
        print("\nEncrypting payload using AES\n")
        encryption = App4AESEncryption().aesEncryption(payload)

        encryptedPayload = None
        with open('encryptedPayload.aes', 'r') as payloadFile:
                encryptedPayload = json.load(payloadFile)
                encryptedPayload = json.dumps(encryptedPayload)

        print("\nSending payload to queue\n")
        rabbit = App4SendRabbitmq()
        rabbit.send_payloadqueue(encryptedPayload)

if __name__ == "__main__":
        main()
```

App4 AES Encryption

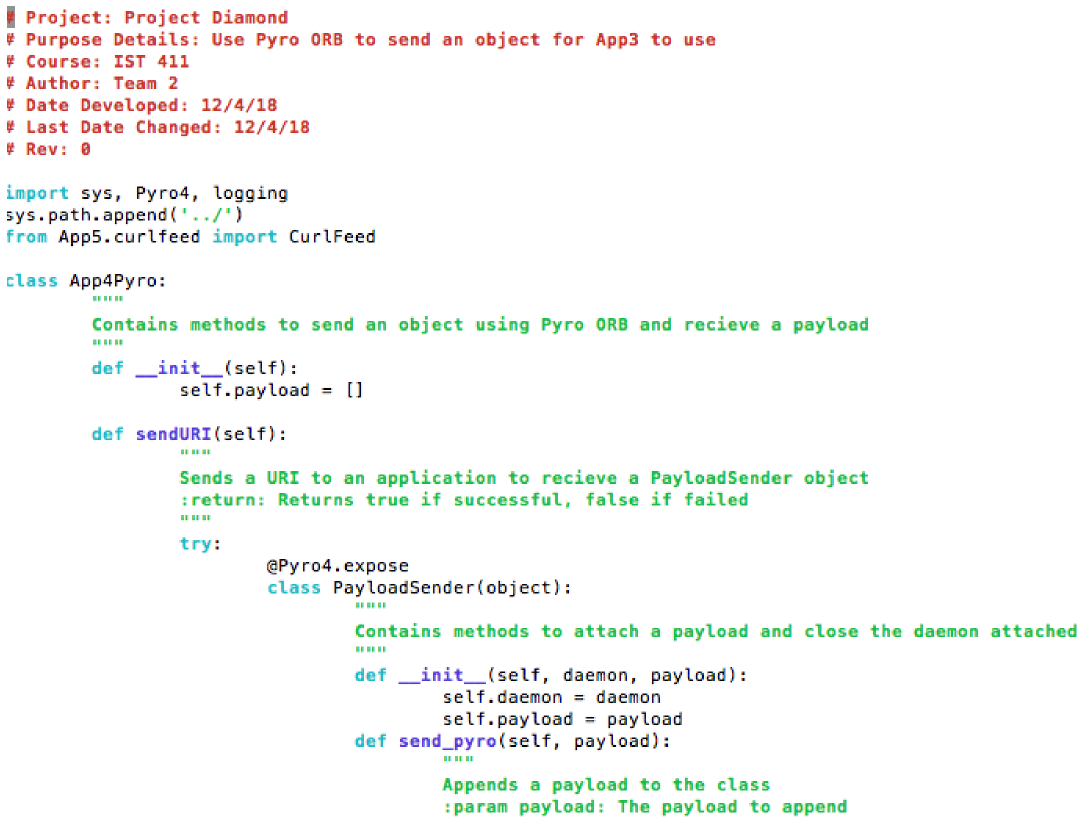


```
  GNU nano 2.9.3                   app4AESEncryption.py

# Project: Project Diamon
# Purpose Details: To take a JSON payload and perform AES encryption
# Course: IST 411
# Author: Team 2
# Date Developed: 11/16/18
# Last Date Changed: 11/16/18
# Rev: 0

import sys, json
from base64 import b64encode
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad
from Crypto.Random import get_random_bytes
sys.path.append('../')
from App5.curlfeed import CurlFeed

class App4AESEncryption:
        """
        A class that contains methods to encrypt a JSON payload
        """
        def __init__(self):
                self.payload = None
                self.encrypytedPayload = None
                self.key = None

        def getPayload(self, payload):
                """
                Obtains a JSON payload from a file
                :return: Returns true if successful, false if failed
                """
                try:
                        print("\nObtaining JSON payload...\n")
                        self.paylod = payload
                        return True
                except:
                        e = sys.exc_info()[0]
                        print("Error: %s" %e)
                        return False
```

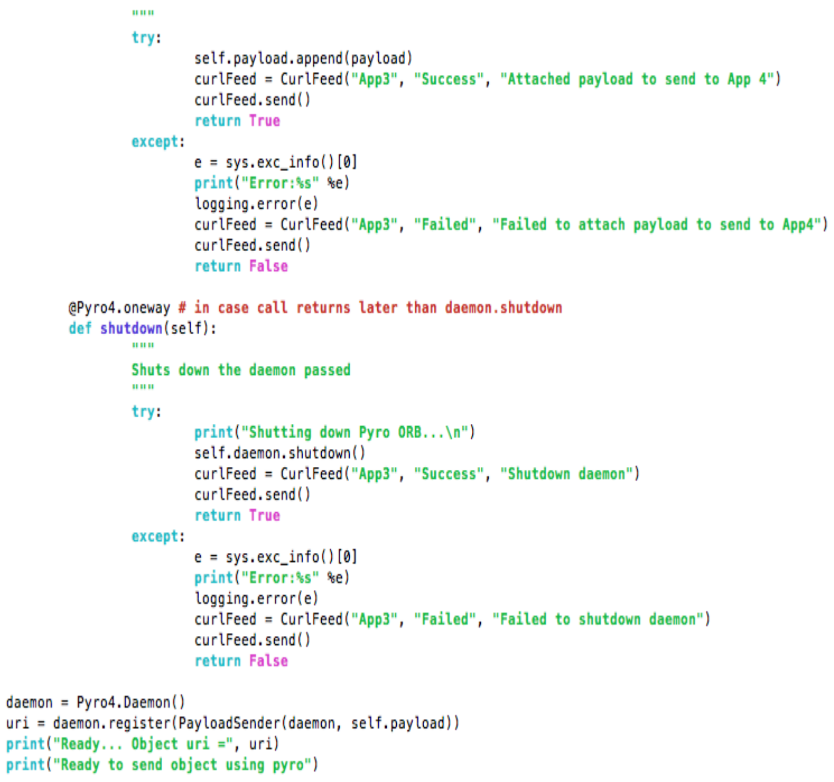


```
        def encrypt(self):
                """
                Takes a payload and performs AES encryption
                :return: Returns true if successful, false if failed
                """
                try:
                        print("\nPerforming encryption on payload...\n")
                        data = json.dumps(self.payload).encode('utf-8')
                        key = (b'thisisakey').ljust(32)[:32]
                        self.key = key
                        cipher = AES.new(key, AES.MODE_CBC)
                        cypher_text_bytes = cipher.encrypt(pad(data, AES.bloc$
                        iv = b64encode(cipher.iv).decode('utf-8')
                        ct = b64encode(cypher_text_bytes).decode('utf-8')
                        result = json.dumps({'iv':iv, 'ciphertext':ct})
                        print(result)
                        self.encryptedPayload = result
                        curlFeed = CurlFeed("App4", "Success", "Encrypted pay$
                        curlFeed.send()
                        return True
                except:
                        e = sys.exc_info()[0]
                        print("Error: %s" %e)
                        curlFeed = CurlFeed("App4", "Failed", "Failed to encr$
                        curlFeed.send()
                        return False


        def savePayload(self):
                """
                Saves an encrpyted payload to a file
                :return: Returns true if successful, false if failed
                """
                print("payload")
                print(self.payload)
                try:
                        print("\nSaving encrypted payload to file...\n")
                        with open('encryptedPayload.aes', 'w') as outFile:
                                outFile.write(self.encryptedPayload)
                        print("Success!")
                        curlFeed = CurlFeed("App4", "Success", "Saved the AES$
                        curlFeed.send()
                        return True
                except:
                        e = sys.exc_info()[0]
                        print("Error: %s" %e)
                        curlFeed = CurlFeed("App4", "Failed", "Failed to save$
                        curlFeed.send()
                        return False

        def aesEncryption(self, payload):
                """
                The main method that calls the methods to encrypt a JSON payl$
                """
                self.getPayload(payload)
                self.encrypt()
                self.savePayload()

if __name__ == "__main__":
        payload = {"test": "something"}
        App4AESEncryption().aesEncryption(payload)
```

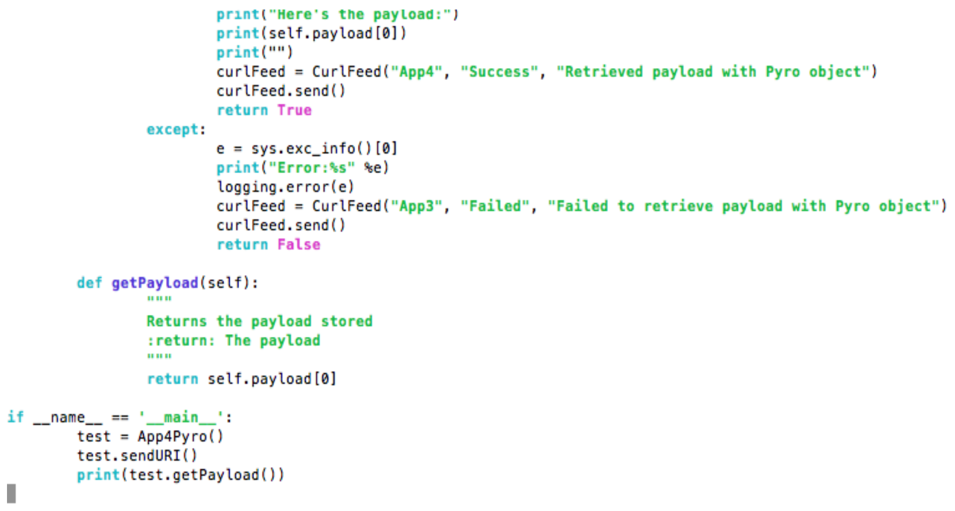
App4 Pyro



```python
# Project: Project Diamond
# Purpose Details: Use Pyro ORB to send an object for App3 to use
# Course: IST 411
# Author: Team 2
# Date Developed: 12/4/18
# Last Date Changed: 12/4/18
# Rev: 0

import sys, Pyro4, logging
sys.path.append('../')
from App5.curlfeed import CurlFeed

class App4Pyro:
    """
    Contains methods to send an object using Pyro ORB and recieve a payload
    """
    def __init__(self):
        self.payload = []

    def sendURI(self):
        """
        Sends a URI to an application to recieve a PayloadSender object
        :return: Returns true if successful, false if failed
        """
        try:
            @Pyro4.expose
            class PayloadSender(object):
                """
                Contains methods to attach a payload and close the daemon attached
                """
                def __init__(self, daemon, payload):
                    self.daemon = daemon
                    self.payload = payload
                def send_pyro(self, payload):
                    """
                    Appends a payload to the class
                    :param payload: The payload to append
```

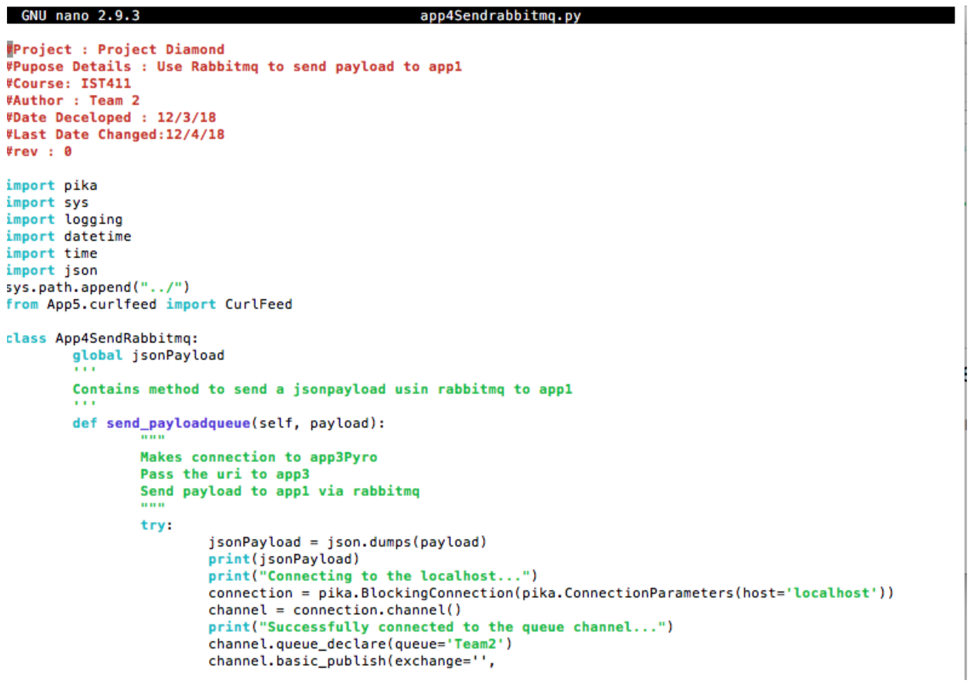


```python
                """
                try:
                    self.payload.append(payload)
                    curlFeed = CurlFeed("App3", "Success", "Attached payload to send to App 4")
                    curlFeed.send()
                    return True
                except:
                    e = sys.exc_info()[0]
                    print("Error:%s" %e)
                    logging.error(e)
                    curlFeed = CurlFeed("App3", "Failed", "Failed to attach payload to send to App4")
                    curlFeed.send()
                    return False

            @Pyro4.oneway # in case call returns later than daemon.shutdown
            def shutdown(self):
                """
                Shuts down the daemon passed
                """
                try:
                    print("Shutting down Pyro ORB...\n")
                    self.daemon.shutdown()
                    curlFeed = CurlFeed("App3", "Success", "Shutdown daemon")
                    curlFeed.send()
                    return True
                except:
                    e = sys.exc_info()[0]
                    print("Error:%s" %e)
                    logging.error(e)
                    curlFeed = CurlFeed("App3", "Failed", "Failed to shutdown daemon")
                    curlFeed.send()
                    return False

        daemon = Pyro4.Daemon()
        uri = daemon.register(PayloadSender(daemon, self.payload))
        print("Ready... Object uri =", uri)
        print("Ready to send object using pyro")
```

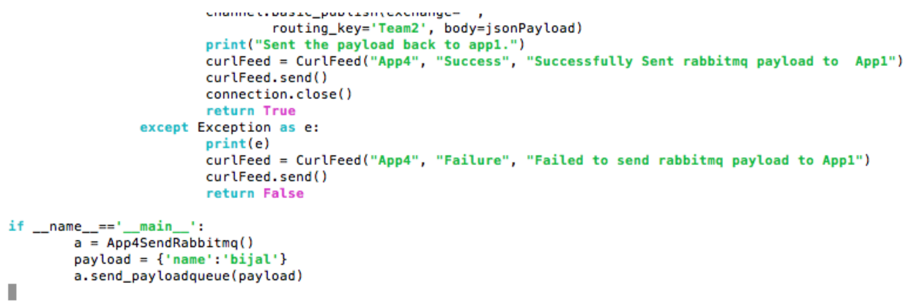


```
                        print("Here's the payload:")
                        print(self.payload[0])
                        print("")
                        curlFeed = CurlFeed("App4", "Success", "Retrieved payload with Pyro object")
                        curlFeed.send()
                        return True
                except:
                        e = sys.exc_info()[0]
                        print("Error:%s" %e)
                        logging.error(e)
                        curlFeed = CurlFeed("App3", "Failed", "Failed to retrieve payload with Pyro object")
                        curlFeed.send()
                        return False

        def getPayload(self):
                """
                Returns the payload stored
                :return: The payload
                """
                return self.payload[0]

if __name__ == '__main__':
        test = App4Pyro()
        test.sendURI()
        print(test.getPayload())
```

App4 Rabbitmq

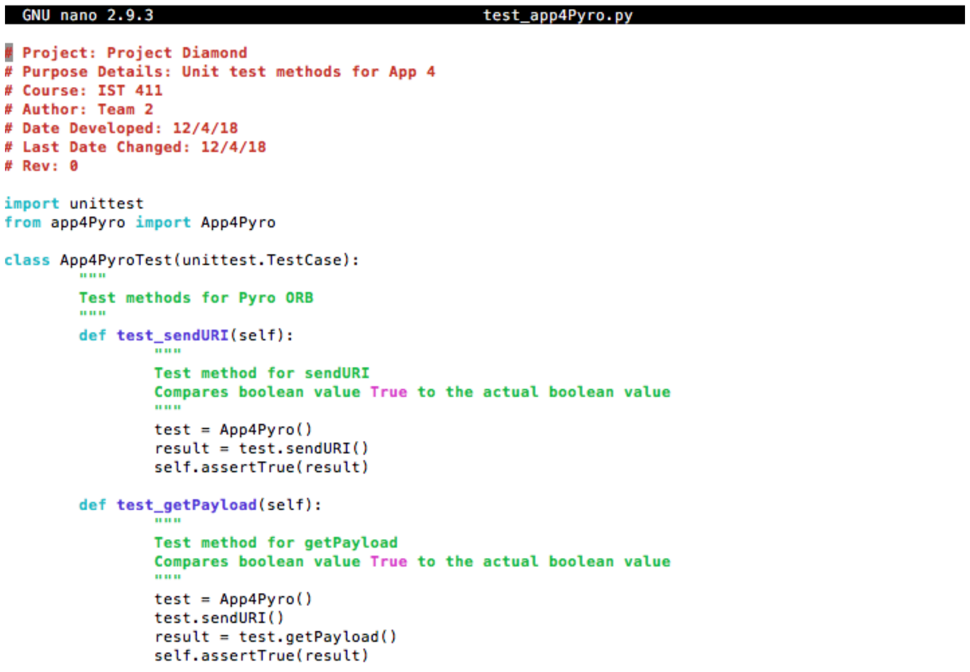


```
GNU nano 2.9.3                                        app4Sendrabbitmq.py

#Project : Project Diamond
#Pupose Details : Use Rabbitmq to send payload to app1
#Course: IST411
#Author : Team 2
#Date Deceloped : 12/3/18
#Last Date Changed:12/4/18
#rev : 0

import pika
import sys
import logging
import datetime
import time
import json
sys.path.append("../")
from App5.curlfeed import CurlFeed

class App4SendRabbitmq:
        global jsonPayload
        '''
        Contains method to send a jsonpayload usin rabbitmq to app1
        '''
        def send_payloadqueue(self, payload):
                """
                Makes connection to app3Pyro
                Pass the uri to app3
                Send payload to app1 via rabbitmq
                """
                try:
                        jsonPayload = json.dumps(payload)
                        print(jsonPayload)
                        print("Connecting to the localhost...")
                        connection = pika.BlockingConnection(pika.ConnectionParameters(host='localhost'))
                        channel = connection.channel()
                        print("Successfully connected to the queue channel...")
                        channel.queue_declare(queue='Team2')
                        channel.basic_publish(exchange='',
```



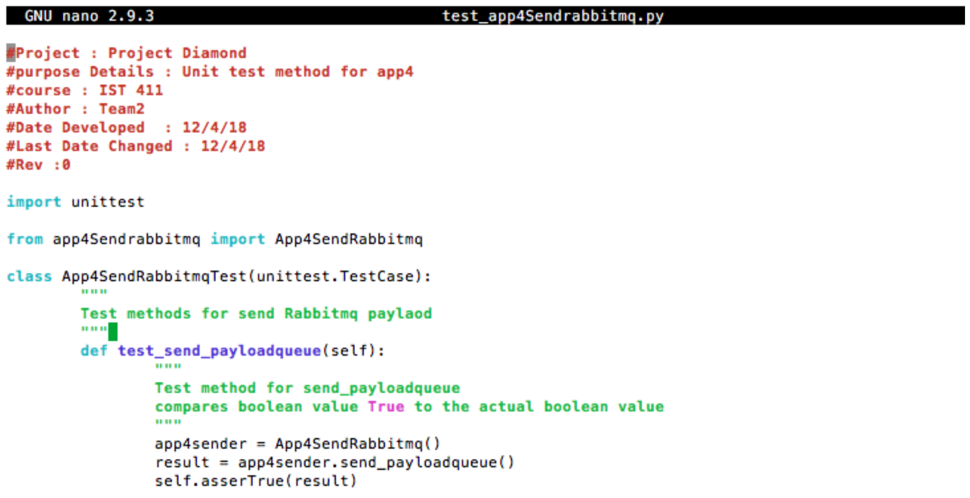
```
                                routing_key='Team2', body=jsonPayload)
                        print("Sent the payload back to app1.")
                        curlFeed = CurlFeed("App4", "Success", "Successfully Sent rabbitmq payload to  App1")
                        curlFeed.send()
                        connection.close()
                        return True
                except Exception as e:
                        print(e)
                        curlFeed = CurlFeed("App4", "Failure", "Failed to send rabbitmq payload to App1")
                        curlFeed.send()
                        return False

if __name__=='__main__':
        a = App4SendRabbitmq()
        payload = {'name':'bijal'}
        a.send_payloadqueue(payload)
```

App4 test_pyro



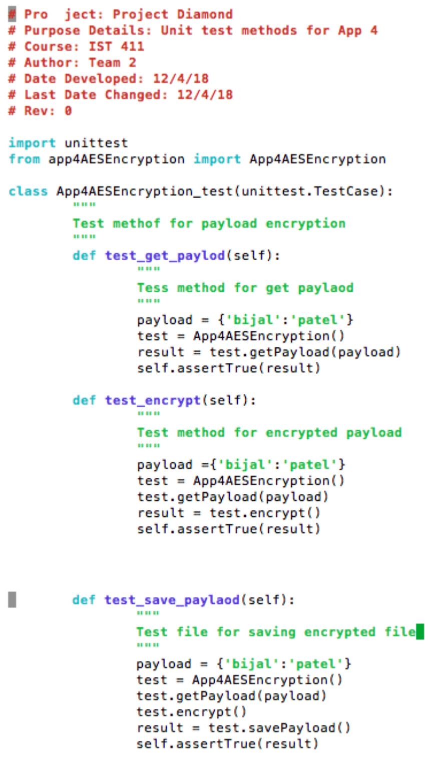
```
 GNU nano 2.9.3                                    test_app4Pyro.py

# Project: Project Diamond
# Purpose Details: Unit test methods for App 4
# Course: IST 411
# Author: Team 2
# Date Developed: 12/4/18
# Last Date Changed: 12/4/18
# Rev: 0

import unittest
from app4Pyro import App4Pyro

class App4PyroTest(unittest.TestCase):
        """
        Test methods for Pyro ORB
        """
        def test_sendURI(self):
                """
                Test method for sendURI
                Compares boolean value True to the actual boolean value
                """
                test = App4Pyro()
                result = test.sendURI()
                self.assertTrue(result)

        def test_getPayload(self):
                """
                Test method for getPayload
                Compares boolean value True to the actual boolean value
                """
                test = App4Pyro()
                test.sendURI()
                result = test.getPayload()
                self.assertTrue(result)
```

App4 test_rabbitmq



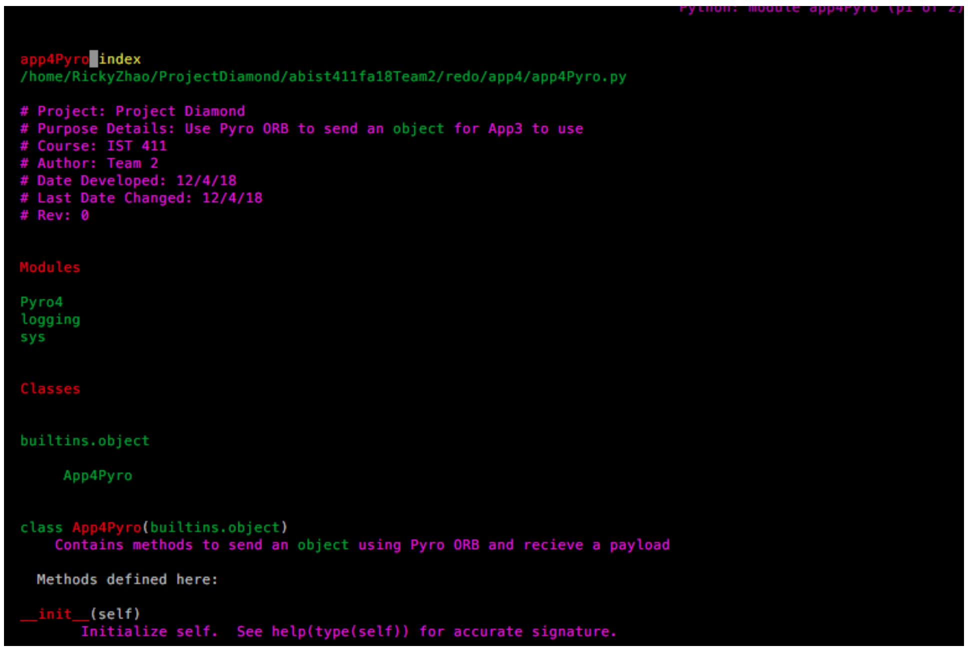
```
 GNU nano 2.9.3                              test_app4Sendrabbitmq.py

#Project : Project Diamond
#purpose Details : Unit test method for app4
#course : IST 411
#Author : Team2
#Date Developed  : 12/4/18
#Last Date Changed : 12/4/18
#Rev :0

import unittest

from app4Sendrabbitmq import App4SendRabbitmq

class App4SendRabbitmqTest(unittest.TestCase):
        """
        Test methods for send Rabbitmq paylaod
        """
        def test_send_payloadqueue(self):
                """
                Test method for send_payloadqueue
                compares boolean value True to the actual boolean value
                """
                app4sender = App4SendRabbitmq()
                result = app4sender.send_payloadqueue()
                self.asserTrue(result)
```

App4 AESEncryption Unittesting

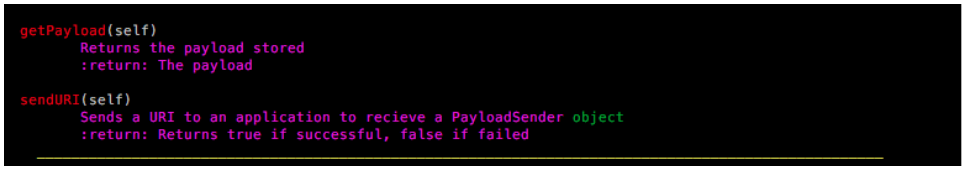


```
# Pro  ject: Project Diamond
# Purpose Details: Unit test methods for App 4
# Course: IST 411
# Author: Team 2
# Date Developed: 12/4/18
# Last Date Changed: 12/4/18
# Rev: 0

import unittest
from app4AESEncryption import App4AESEncryption

class App4AESEncryption_test(unittest.TestCase):
        """
        Test methof for payload encryption
        """
        def test_get_paylod(self):
                """
                Tess method for get paylaod
                """
                payload = {'bijal':'patel'}
                test = App4AESEncryption()
                result = test.getPayload(payload)
                self.assertTrue(result)

        def test_encrypt(self):
                """
                Test method for encrypted payload
                """
                payload ={'bijal':'patel'}
                test = App4AESEncryption()
                test.getPayload(payload)
                result = test.encrypt()
                self.assertTrue(result)


        def test_save_paylaod(self):
                """
                Test file for saving encrypted file
                """
                payload = {'bijal':'patel'}
                test = App4AESEncryption()
                test.getPayload(payload)
                test.encrypt()
                result = test.savePayload()
                self.assertTrue(result)
```

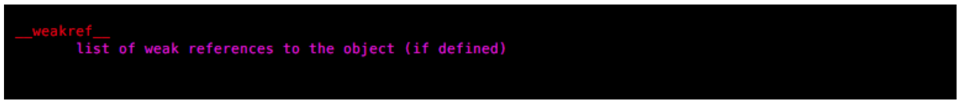
App4 pyro html



```
                                                                       Python: module app4Pyro (p1 of 2)

 app4Pyro index
 /home/RickyZhao/ProjectDiamond/abist411fa18Team2/redo/app4/app4Pyro.py

 # Project: Project Diamond
 # Purpose Details: Use Pyro ORB to send an object for App3 to use
 # Course: IST 411
 # Author: Team 2
 # Date Developed: 12/4/18
 # Last Date Changed: 12/4/18
 # Rev: 0


 Modules

 Pyro4
 logging
 sys


 Classes


 builtins.object

      App4Pyro


 class App4Pyro(builtins.object)
     Contains methods to send an object using Pyro ORB and recieve a payload

   Methods defined here:

 __init__(self)
        Initialize self.  See help(type(self)) for accurate signature.
```

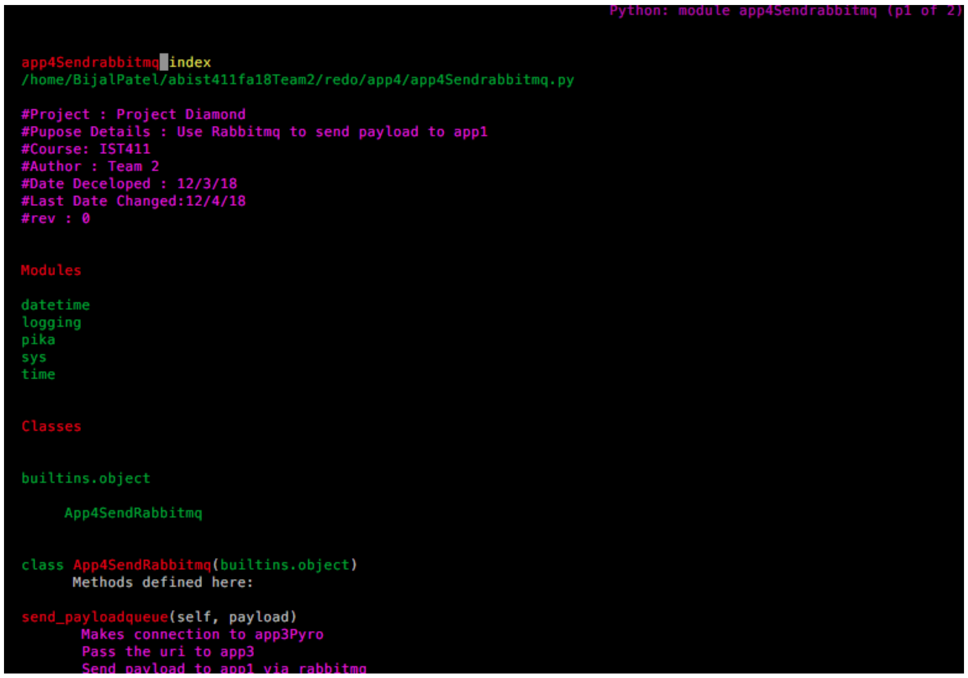


```
getPayload(self)
       Returns the payload stored
       :return: The payload

sendURI(self)
       Sends a URI to an application to recieve a PayloadSender object
       :return: Returns true if successful, false if failed
```

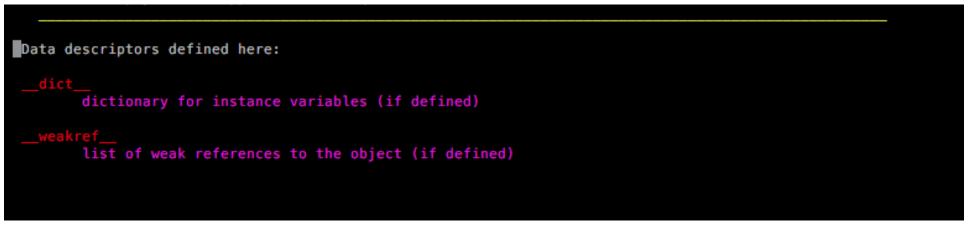


```
__weakref__
       list of weak references to the object (if defined)
```

App4 Rabbitmq html

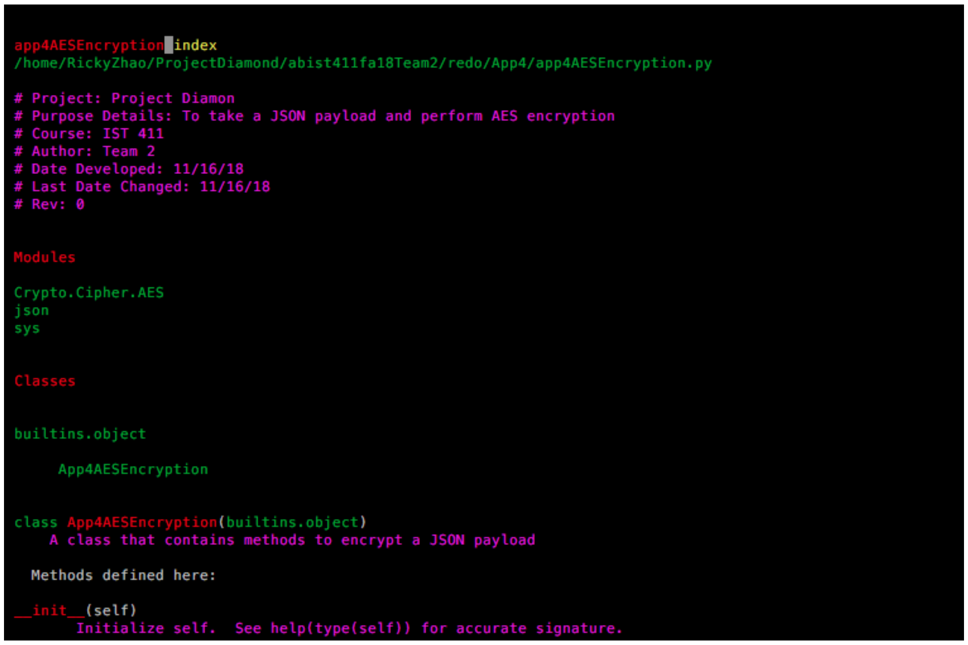


```
                                                                    Python: module app4Sendrabbitmq (p1 of 2)

app4Sendrabbitmq index
/home/BijalPatel/abist411fa18Team2/redo/app4/app4Sendrabbitmq.py

#Project : Project Diamond
#Pupose Details : Use Rabbitmq to send payload to app1
#Course: IST411
#Author : Team 2
#Date Deceloped : 12/3/18
#Last Date Changed:12/4/18
#rev : 0


Modules

datetime
logging
pika
sys
time


Classes


builtins.object

     App4SendRabbitmq


class App4SendRabbitmq(builtins.object)
      Methods defined here:

send_payloadqueue(self, payload)
       Makes connection to app3Pyro
       Pass the uri to app3
       Send payload to app1 via rabbitmq
```

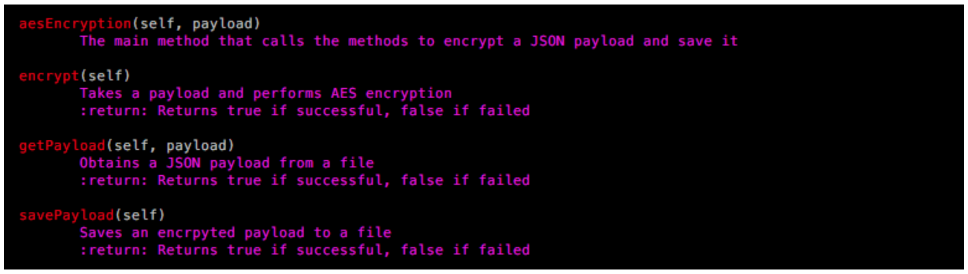


```
Data descriptors defined here:

__dict__
       dictionary for instance variables (if defined)

__weakref__
       list of weak references to the object (if defined)
```

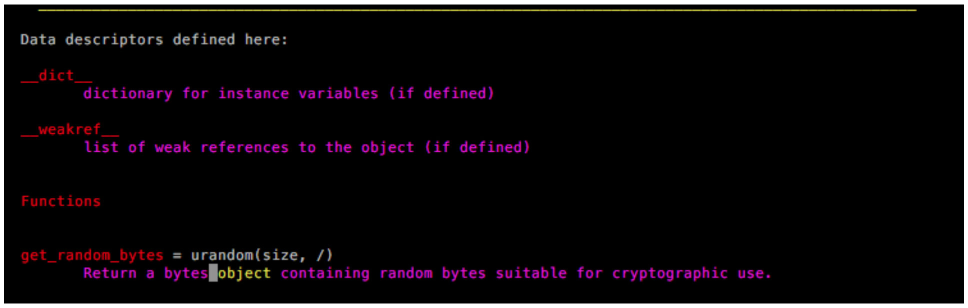
App4 AESencryption.html



```
app4AESEncryption index
/home/RickyZhao/ProjectDiamond/abist411fa18Team2/redo/App4/app4AESEncryption.py

# Project: Project Diamon
# Purpose Details: To take a JSON payload and perform AES encryption
# Course: IST 411
# Author: Team 2
# Date Developed: 11/16/18
# Last Date Changed: 11/16/18
# Rev: 0


Modules

Crypto.Cipher.AES
json
sys


Classes


builtins.object

     App4AESEncryption


class App4AESEncryption(builtins.object)
    A class that contains methods to encrypt a JSON payload

  Methods defined here:

__init__(self)
       Initialize self.  See help(type(self)) for accurate signature.
```

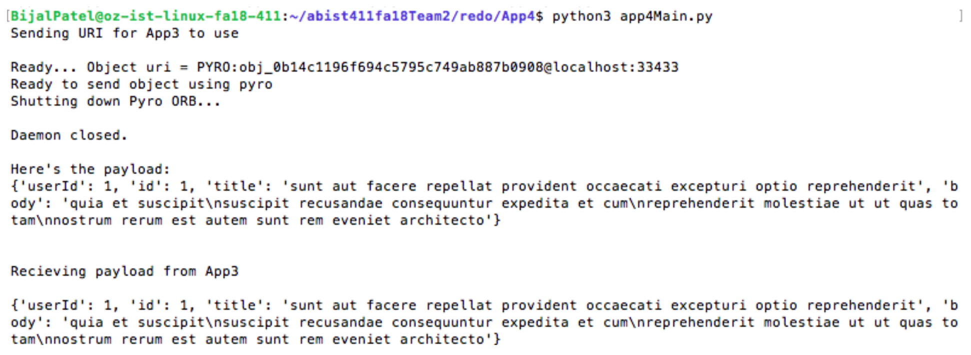


```
aesEncryption(self, payload)
       The main method that calls the methods to encrypt a JSON payload and save it

encrypt(self)
       Takes a payload and performs AES encryption
       :return: Returns true if successful, false if failed

getPayload(self, payload)
       Obtains a JSON payload from a file
       :return: Returns true if successful, false if failed

savePayload(self)
       Saves an encrpyted payload to a file
       :return: Returns true if successful, false if failed
```

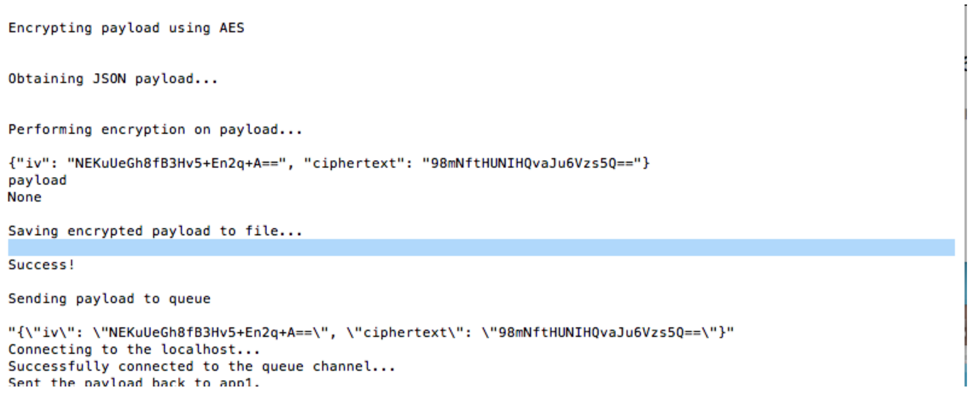


```
Data descriptors defined here:

__dict__
       dictionary for instance variables (if defined)

__weakref__
       list of weak references to the object (if defined)


Functions


get_random_bytes = urandom(size, /)
       Return a bytes object containing random bytes suitable for cryptographic use.
```

App4 Main running

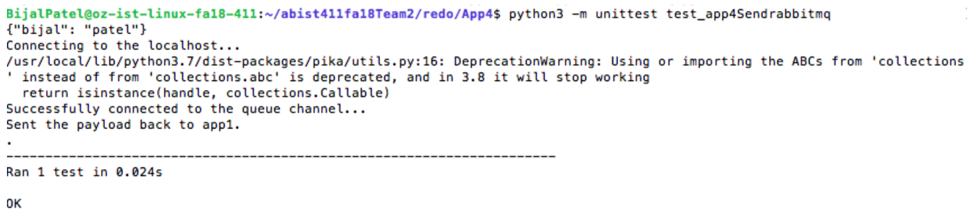


```
BijalPatel@oz-ist-linux-fa18-411:~/abist411fa18Team2/redo/App4$ python3 app4Main.py
Sending URI for App3 to use

Ready... Object uri = PYRO:obj_0b14c1196f694c5795c749ab887b0908@localhost:33433
Ready to send object using pyro
Shutting down Pyro ORB...

Daemon closed.

Here's the payload:
{'userId': 1, 'id': 1, 'title': 'sunt aut facere repellat provident occaecati excepturi optio reprehenderit', 'b
ody': 'quia et suscipit\nsuscipit recusandae consequuntur expedita et cum\nreprehenderit molestiae ut ut quas to
tam\nnostrum rerum est autem sunt rem eveniet architecto'}


Recieving payload from App3

{'userId': 1, 'id': 1, 'title': 'sunt aut facere repellat provident occaecati excepturi optio reprehenderit', 'b
ody': 'quia et suscipit\nsuscipit recusandae consequuntur expedita et cum\nreprehenderit molestiae ut ut quas to
tam\nnostrum rerum est autem sunt rem eveniet architecto'}
```



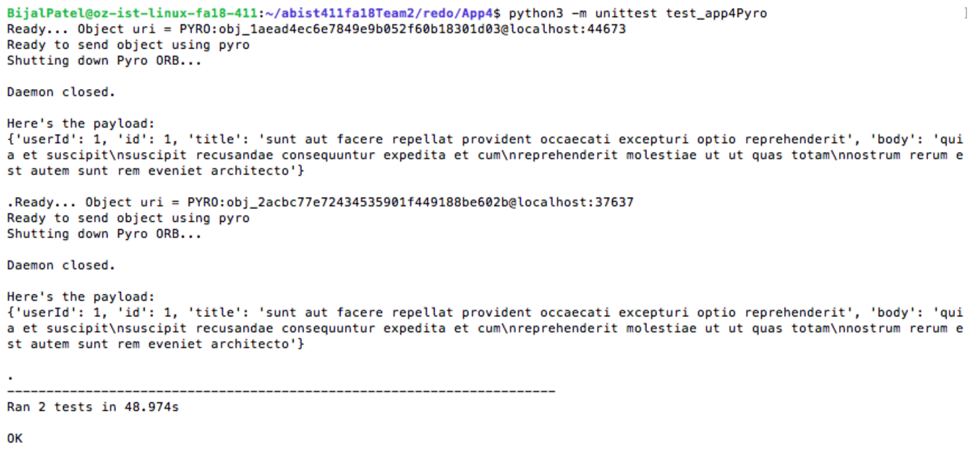
```
Encrypting payload using AES


Obtaining JSON payload...


Performing encryption on payload...

{"iv": "NEKuUeGh8fB3Hv5+En2q+A==", "ciphertext": "98mNftHUNIHQvaJu6Vzs5Q=="}
payload
None

Saving encrypted payload to file...

Success!

Sending payload to queue

"{\"iv\": \"NEKuUeGh8fB3Hv5+En2q+A==\", \"ciphertext\": \"98mNftHUNIHQvaJu6Vzs5Q==\"}"
Connecting to the localhost...
Successfully connected to the queue channel...
Sent the payload back to app1.
```

App4 Unittest for rabbitmq



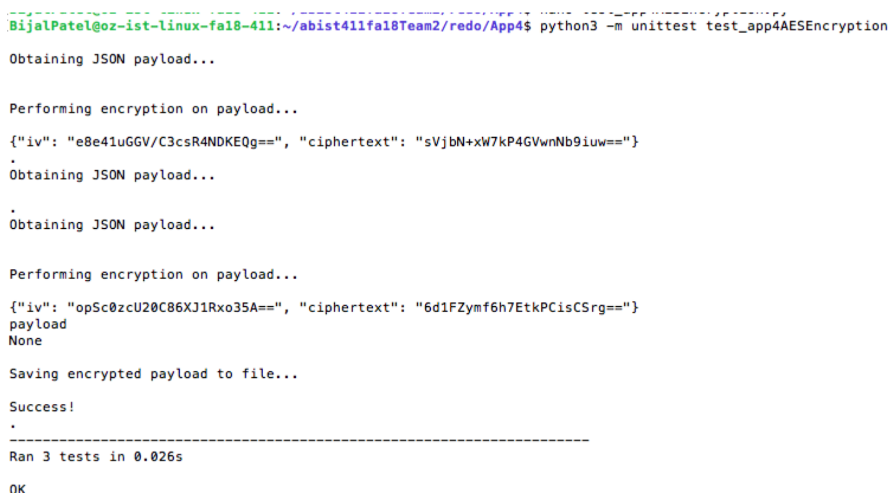
```
BijalPatel@oz-ist-linux-fa18-411:~/abist411fa18Team2/redo/App4$ python3 -m unittest test_app4Sendrabbitmq
{"bijal": "patel"}
Connecting to the localhost...
/usr/local/lib/python3.7/dist-packages/pika/utils.py:16: DeprecationWarning: Using or importing the ABCs from 'collections
' instead of from 'collections.abc' is deprecated, and in 3.8 it will stop working
  return isinstance(handle, collections.Callable)
Successfully connected to the queue channel...
Sent the payload back to app1.
.
----------------------------------------------------------------------
Ran 1 test in 0.024s

OK
```

App4 unittest for Pyro



```
BijalPatel@oz-ist-linux-fa18-411:~/abist411fa18Team2/redo/App4$ python3 -m unittest test_app4Pyro
Ready... Object uri = PYRO:obj_1aead4ec6e7849e9b052f60b18301d03@localhost:44673
Ready to send object using pyro
Shutting down Pyro ORB...

Daemon closed.

Here's the payload:
{'userId': 1, 'id': 1, 'title': 'sunt aut facere repellat provident occaecati excepturi optio reprehenderit', 'body': 'qui
a et suscipit\nsuscipit recusandae consequuntur expedita et cum\nreprehenderit molestiae ut ut quas totam\nnostrum rerum e
st autem sunt rem eveniet architecto'}

.Ready... Object uri = PYRO:obj_2acbc77e72434535901f449188be602b@localhost:37637
Ready to send object using pyro
Shutting down Pyro ORB...

Daemon closed.

Here's the payload:
{'userId': 1, 'id': 1, 'title': 'sunt aut facere repellat provident occaecati excepturi optio reprehenderit', 'body': 'qui
a et suscipit\nsuscipit recusandae consequuntur expedita et cum\nreprehenderit molestiae ut ut quas totam\nnostrum rerum e
st autem sunt rem eveniet architecto'}

.
----------------------------------------------------------------------
Ran 2 tests in 48.974s

OK
```

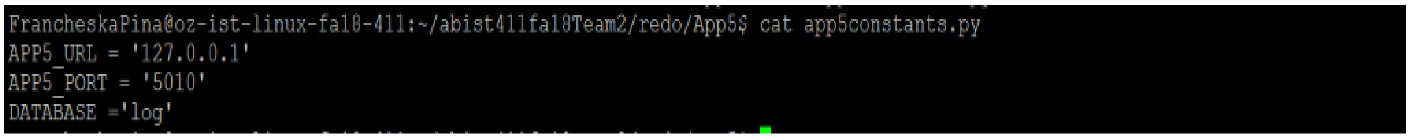
App4 Unittest for AESencryption



```
BijalPatel@oz-ist-linux-fa18-411:~/abist411fa18Team2/redo/App4$ python3 -m unittest test_app4AESEncryption

Obtaining JSON payload...


Performing encryption on payload...

{"iv": "e8e41uGGV/C3csR4NDKEQg==", "ciphertext": "sVjbN+xW7kP4GVwnNb9iuw=="}
.
Obtaining JSON payload...

.
Obtaining JSON payload...


Performing encryption on payload...

{"iv": "opSc0zcU20C86XJ1Rxo35A==", "ciphertext": "6d1FZymf6h7EtkPCisCSrg=="}
payload
None

Saving encrypted payload to file...

Success!
.
----------------------------------------------------------------------
Ran 3 tests in 0.026s

OK
```

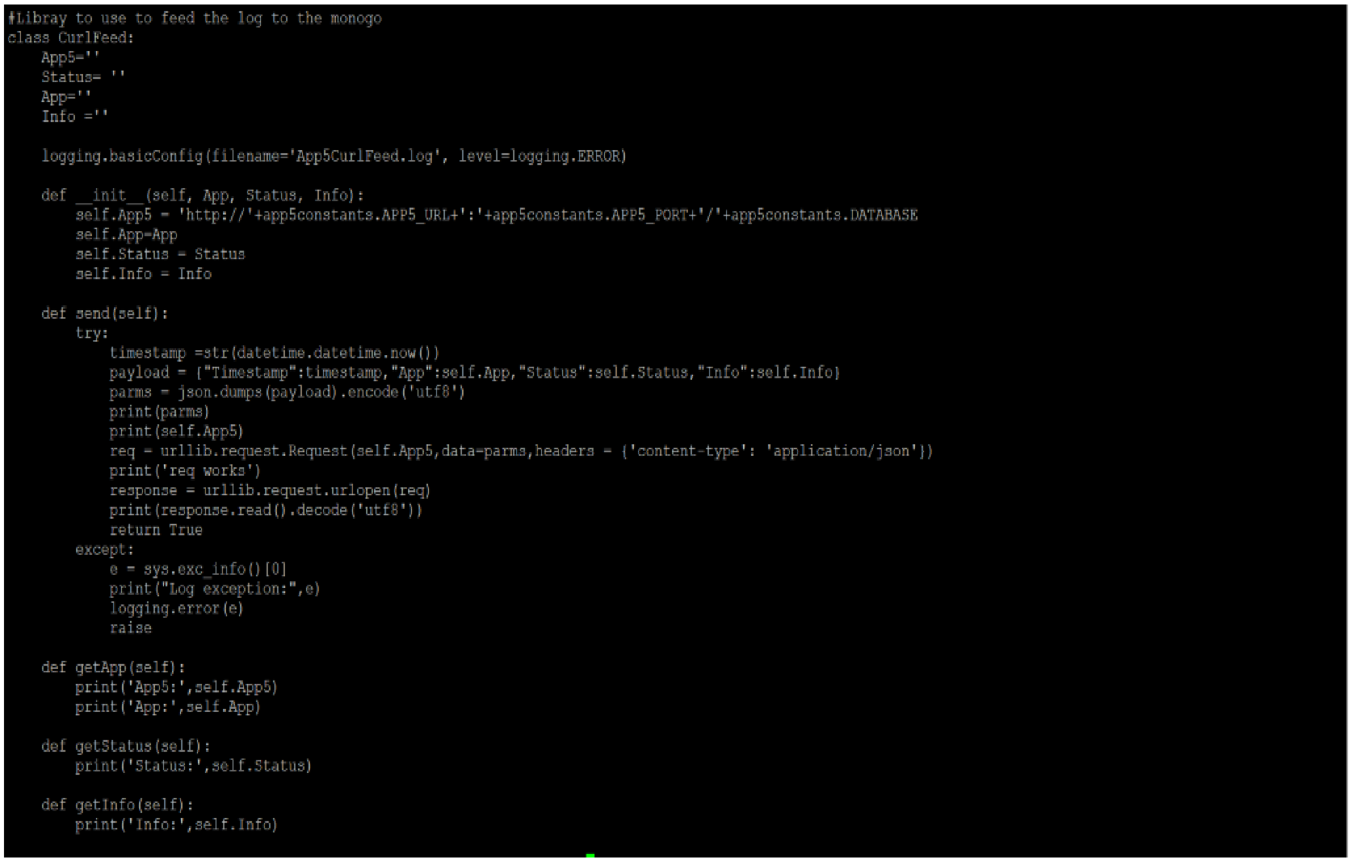
App 5

App5 constants



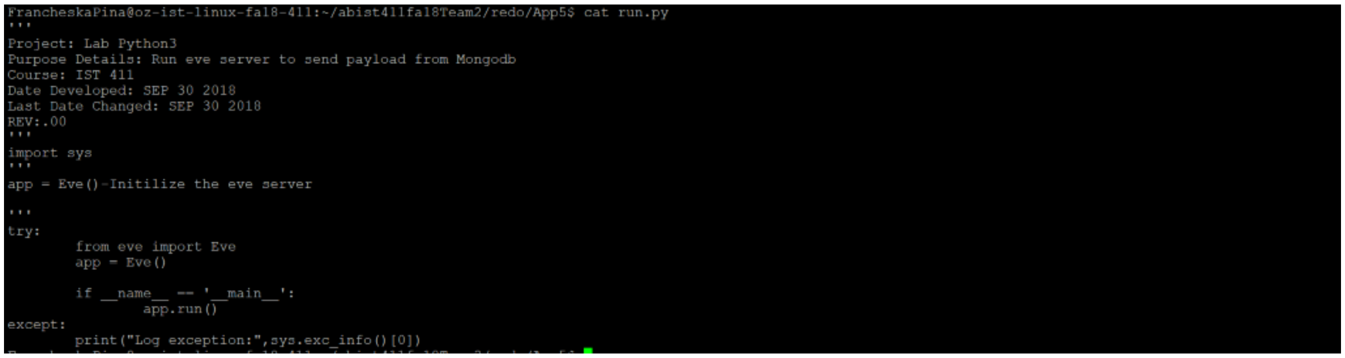
```
FrancheskaPina@oz-ist-linux-fa18-411:~/abist411fa18Team2/redo/App5$ cat app5constants.py
APP5_URL = '127.0.0.1'
APP5_PORT = '5010'
DATABASE ='log'
```

App5 curlfeed



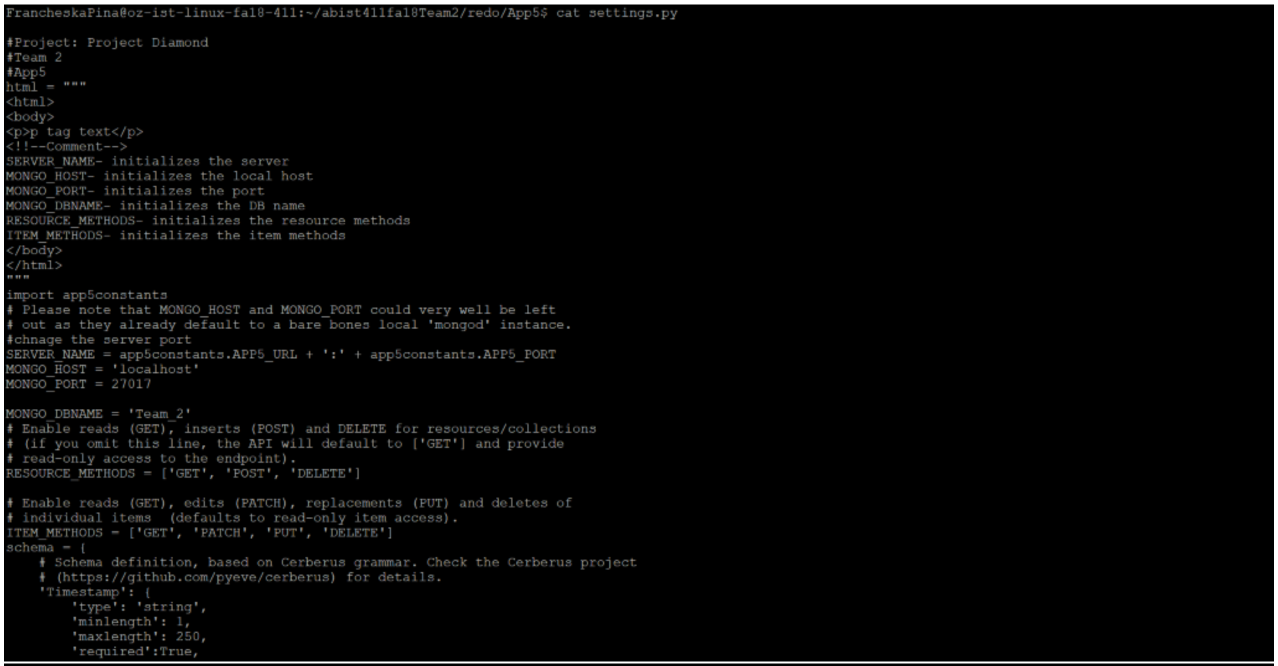
```
#Libray to use to feed the log to the monogo
class CurlFeed:
    App5=''
    Status= ''
    App=''
    Info =''

    logging.basicConfig(filename='App5CurlFeed.log', level=logging.ERROR)

    def __init__(self, App, Status, Info):
        self.App5 = 'http://'+app5constants.APP5_URL+':'+app5constants.APP5_PORT+'/'+app5constants.DATABASE
        self.App=App
        self.Status = Status
        self.Info = Info

    def send(self):
        try:
            timestamp =str(datetime.datetime.now())
            payload = {"Timestamp":timestamp,"App":self.App,"Status":self.Status,"Info":self.Info}
            parms = json.dumps(payload).encode('utf8')
            print(parms)
            print(self.App5)
            req = urllib.request.Request(self.App5,data=parms,headers = {'content-type': 'application/json'})
            print('req works')
            response = urllib.request.urlopen(req)
            print(response.read().decode('utf8'))
            return True
        except:
            e = sys.exc_info()[0]
            print("Log exception:",e)
            logging.error(e)
            raise

    def getApp(self):
        print('App5:',self.App5)
        print('App:',self.App)

    def getStatus(self):
        print('Status:',self.Status)

    def getInfo(self):
        print('Info:',self.Info)
```

App5 run class

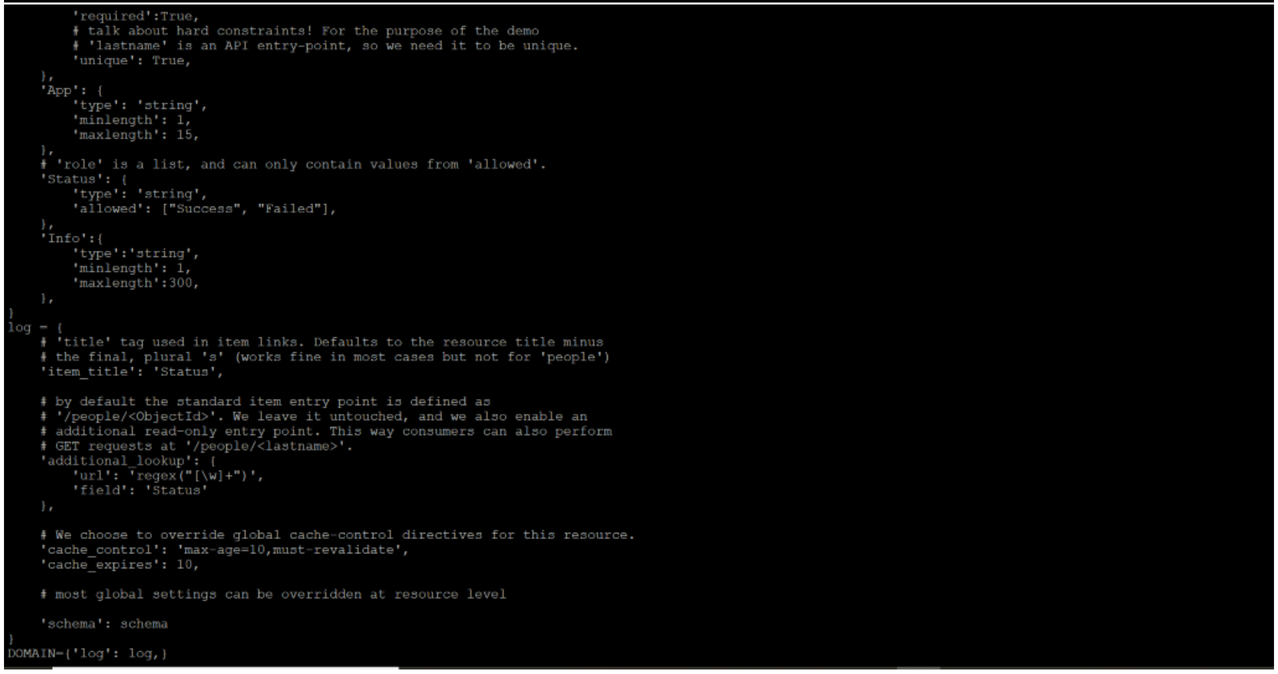


```
FrancheskaPina@oz-ist-linux-fa18-411:~/abist411fa18Team2/redo/App5$ cat run.py
'''
Project: Lab Python3
Purpose Details: Run eve server to send payload from Mongodb
Course: IST 411
Date Developed: SEP 30 2018
Last Date Changed: SEP 30 2018
REV:.00
'''
import sys
'''
app = Eve()-Initilize the eve server

'''
try:
        from eve import Eve
        app = Eve()

        if __name__ == '__main__':
                app.run()
except:
        print("Log exception:",sys.exc_info()[0])
```

App5 settings class

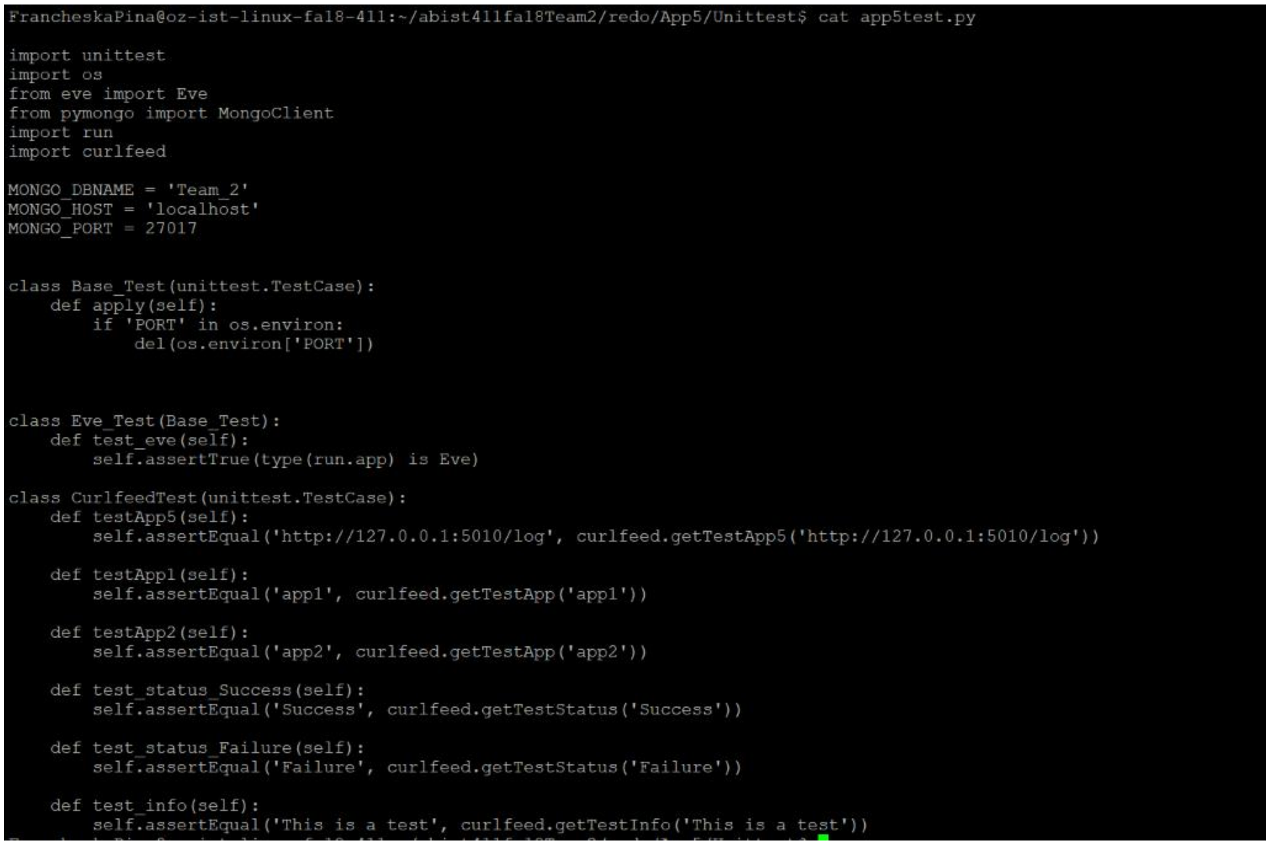


```
FrancheskaPina@oz-ist-linux-fa18-411:~/abist411fa18Team2/redo/App5$ cat settings.py

#Project: Project Diamond
#Team 2
#App5
html = """
<html>
<body>
<p>p tag text</p>
<!--Comment-->
SERVER_NAME- initializes the server
MONGO_HOST- initializes the local host
MONGO_PORT- initializes the port
MONGO_DBNAME- initializes the DB name
RESOURCE_METHODS- initializes the resource methods
ITEM_METHODS- initializes the item methods
</body>
</html>
"""
import app5constants
# Please note that MONGO_HOST and MONGO_PORT could very well be left
# out as they already default to a bare bones local 'mongod' instance.
#chnage the server port
SERVER_NAME = app5constants.APP5_URL + ':' + app5constants.APP5_PORT
MONGO_HOST = 'localhost'
MONGO_PORT = 27017

MONGO_DBNAME = 'Team_2'
# Enable reads (GET), inserts (POST) and DELETE for resources/collections
# (if you omit this line, the API will default to ['GET'] and provide
# read-only access to the endpoint).
RESOURCE_METHODS = ['GET', 'POST', 'DELETE']

# Enable reads (GET), edits (PATCH), replacements (PUT) and deletes of
# individual items  (defaults to read-only item access).
ITEM_METHODS = ['GET', 'PATCH', 'PUT', 'DELETE']
schema = {
    # Schema definition, based on Cerberus grammar. Check the Cerberus project
    # (https://github.com/pyeve/cerberus) for details.
    'Timestamp': {
        'type': 'string',
        'minlength': 1,
        'maxlength': 250,
        'required':True,
```



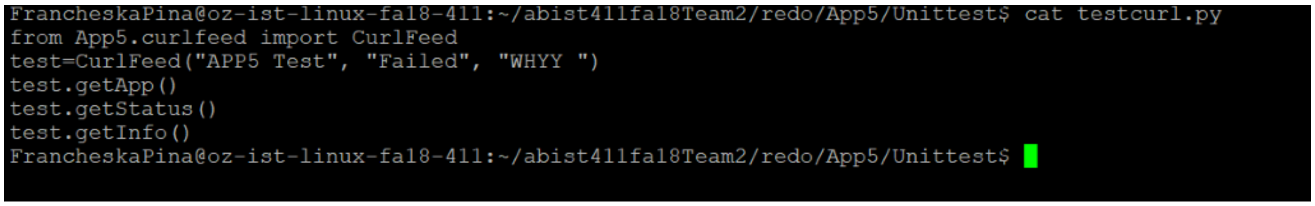
```
        'required':True,
        # talk about hard constraints! For the purpose of the demo
        # 'lastname' is an API entry-point, so we need it to be unique.
        'unique': True,
    },
    'App': {
        'type': 'string',
        'minlength': 1,
        'maxlength': 15,
    },
    # 'role' is a list, and can only contain values from 'allowed'.
    'Status': {
        'type': 'string',
        'allowed': ["Success", "Failed"],
    },
    'Info':{
        'type':'string',
        'minlength': 1,
        'maxlength':300,
    },
}
log = {
    # 'title' tag used in item links. Defaults to the resource title minus
    # the final, plural 's' (works fine in most cases but not for 'people')
    'item_title': 'Status',

    # by default the standard item entry point is defined as
    # '/people/<ObjectId>'. We leave it untouched, and we also enable an
    # additional read-only entry point. This way consumers can also perform
    # GET requests at '/people/<lastname>'.
    'additional_lookup': {
        'url': 'regex("[\w]+")',
        'field': 'Status'
    },

    # We choose to override global cache-control directives for this resource.
    'cache_control': 'max-age=10,must-revalidate',
    'cache_expires': 10,

    # most global settings can be overridden at resource level

    'schema': schema
}
DOMAIN={'log': log,}
```

App5 unittest



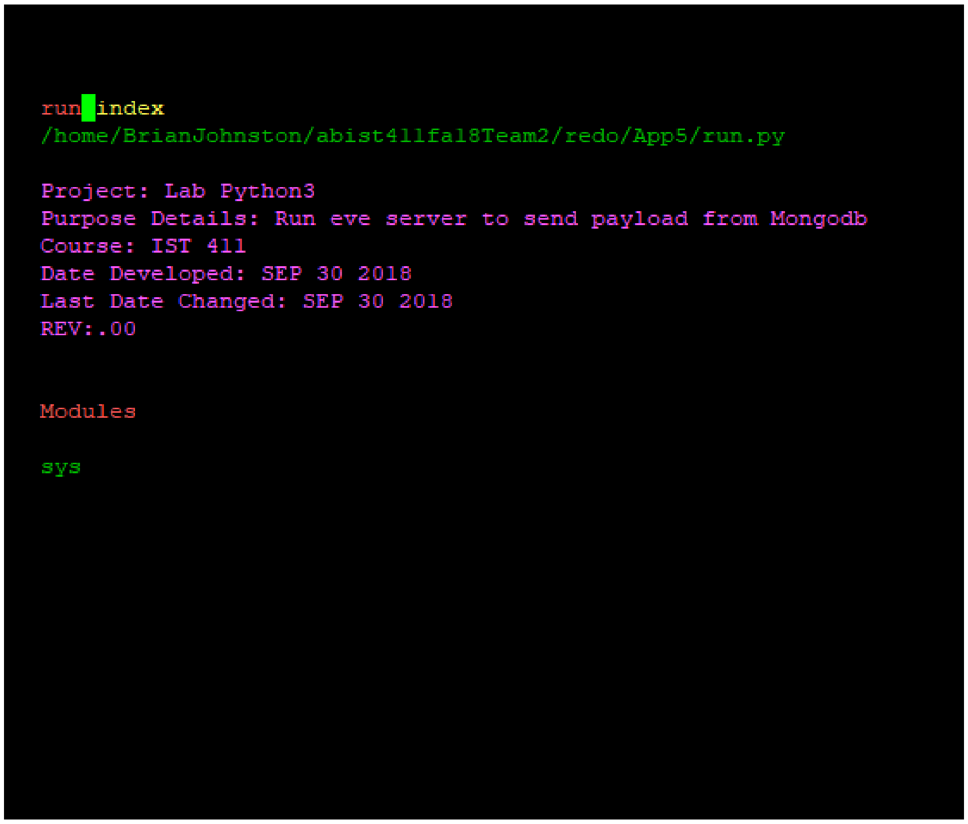
```
FrancheskaPina@oz-ist-linux-fa18-411:~/abist411fa18Team2/redo/App5/Unittest$ cat app5test.py

import unittest
import os
from eve import Eve
from pymongo import MongoClient
import run
import curlfeed

MONGO_DBNAME = 'Team_2'
MONGO_HOST = 'localhost'
MONGO_PORT = 27017


class Base_Test(unittest.TestCase):
    def apply(self):
        if 'PORT' in os.environ:
            del(os.environ['PORT'])



class Eve_Test(Base_Test):
    def test_eve(self):
        self.assertTrue(type(run.app) is Eve)

class CurlfeedTest(unittest.TestCase):
    def testApp5(self):
        self.assertEqual('http://127.0.0.1:5010/log', curlfeed.getTestApp5('http://127.0.0.1:5010/log'))

    def testApp1(self):
        self.assertEqual('app1', curlfeed.getTestApp('app1'))

    def testApp2(self):
        self.assertEqual('app2', curlfeed.getTestApp('app2'))

    def test_status_Success(self):
        self.assertEqual('Success', curlfeed.getTestStatus('Success'))

    def test_status_Failure(self):
        self.assertEqual('Failure', curlfeed.getTestStatus('Failure'))

    def test_info(self):
        self.assertEqual('This is a test', curlfeed.getTestInfo('This is a test'))
```

App5 unit test curl



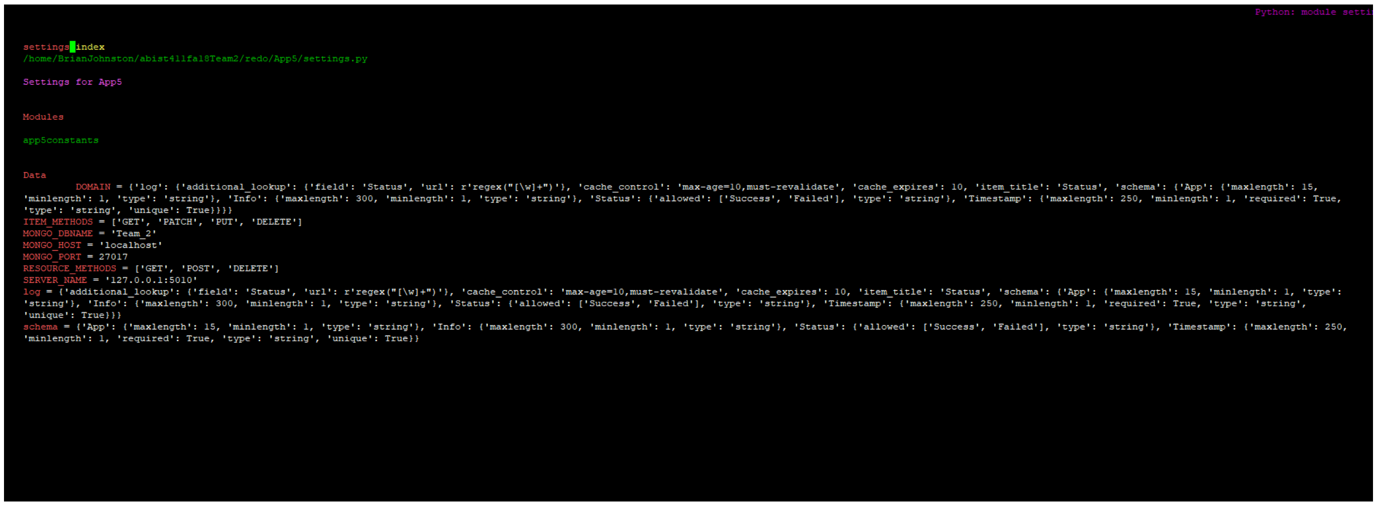
```
FrancheskaPina@oz-ist-linux-fa18-411:~/abist411fa18Team2/redo/App5/Unittest$ cat testcurl.py
from App5.curlfeed import CurlFeed
test=CurlFeed("APP5 Test", "Failed", "WHYY ")
test.getApp()
test.getStatus()
test.getInfo()
FrancheskaPina@oz-ist-linux-fa18-411:~/abist411fa18Team2/redo/App5/Unittest$
```

App5 run HTML



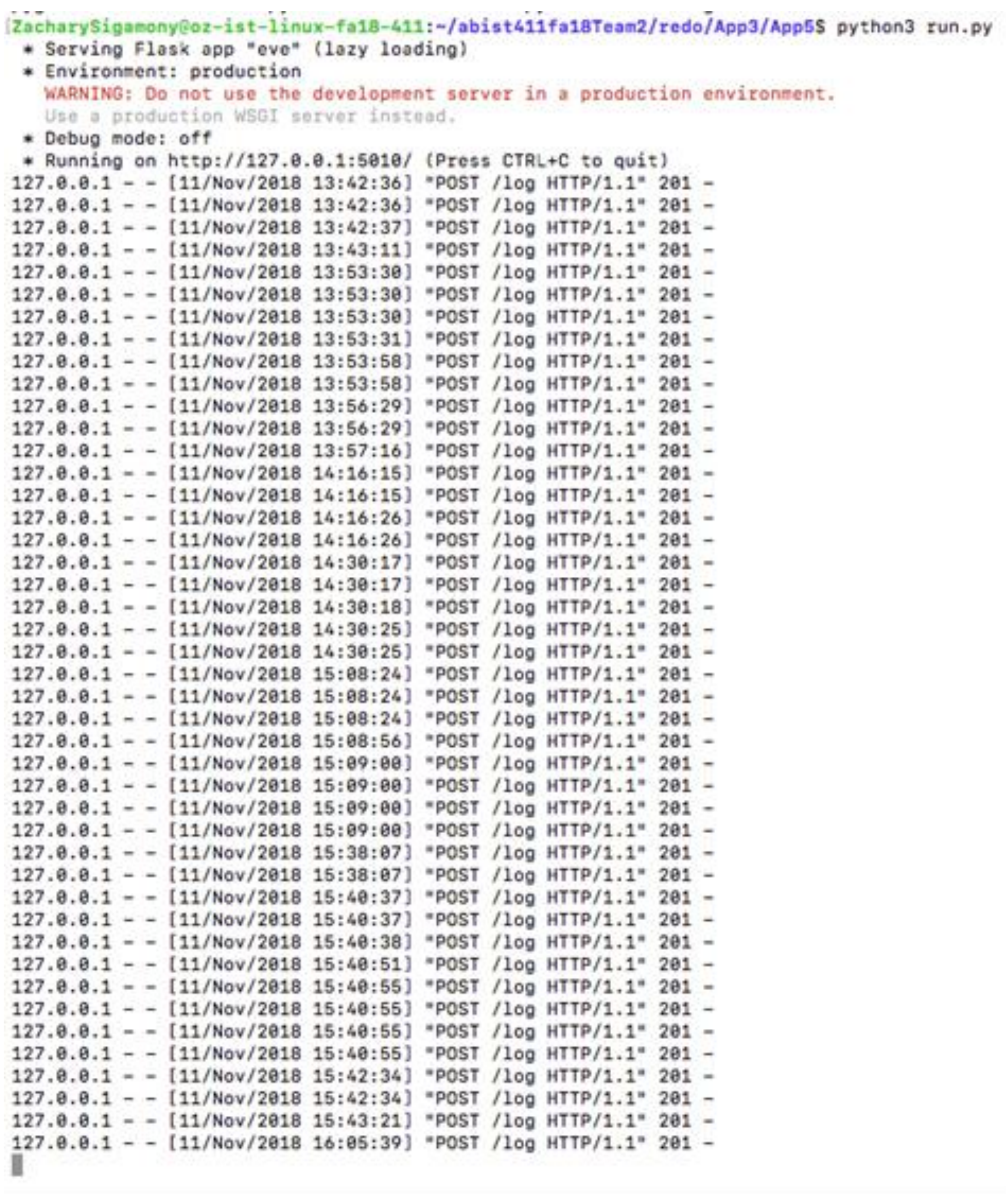
```
run index
/home/BrianJohnston/abist411fa18Team2/redo/App5/run.py

Project: Lab Python3
Purpose Details: Run eve server to send payload from Mongodb
Course: IST 411
Date Developed: SEP 30 2018
Last Date Changed: SEP 30 2018
REV:.00


Modules

sys
```

App5 settings HTML



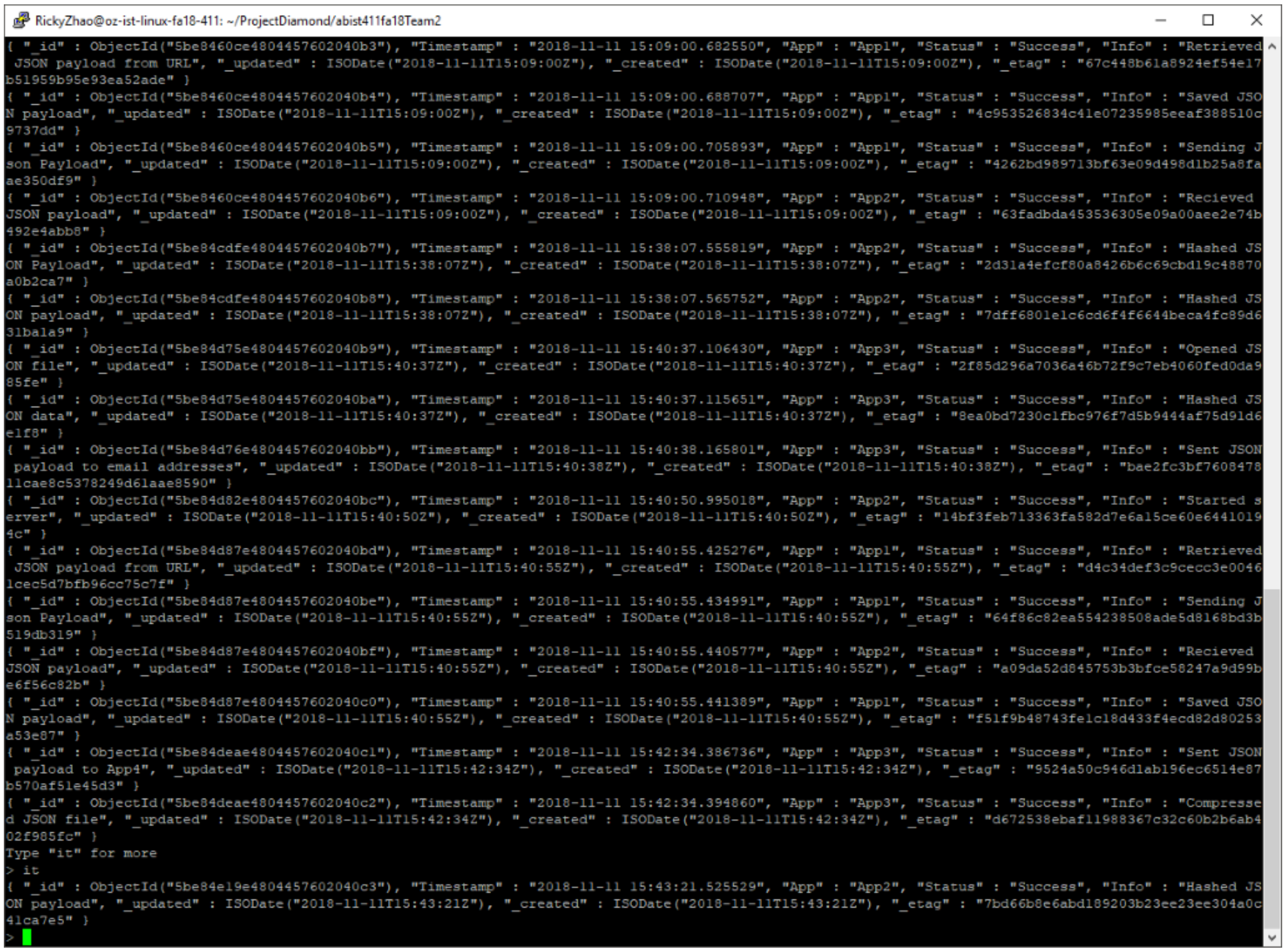
```
                                                                                                    Python: module setti

settings index
/home/BrianJohnston/abist411fa18Team2/redo/App5/settings.py

Settings for App5


Modules

app5constants


Data
        DOMAIN = {'log': {'additional_lookup': {'field': 'Status', 'url': r'regex("[\w]+")'}, 'cache_control': 'max-age=10,must-revalidate', 'cache_expires': 10, 'item_title': 'Status', 'schema': {'App': {'maxlength': 15,
'minlength': 1, 'type': 'string'}, 'Info': {'maxlength': 300, 'minlength': 1, 'type': 'string'}, 'Status': {'allowed': ['Success', 'Failed'], 'type': 'string'}, 'Timestamp': {'maxlength': 250, 'minlength': 1, 'required': True,
'type': 'string', 'unique': True}}}}
ITEM_METHODS = ['GET', 'PATCH', 'PUT', 'DELETE']
MONGO_DBNAME = 'Team_2'
MONGO_HOST = 'localhost'
MONGO_PORT = 27017
RESOURCE_METHODS = ['GET', 'POST', 'DELETE']
SERVER_NAME = '127.0.0.1:5010'
log = {'additional_lookup': {'field': 'Status', 'url': r'regex("[\w]+")'}, 'cache_control': 'max-age=10,must-revalidate', 'cache_expires': 10, 'item_title': 'Status', 'schema': {'App': {'maxlength': 15, 'minlength': 1, 'type':
'string'}, 'Info': {'maxlength': 300, 'minlength': 1, 'type': 'string'}, 'Status': {'allowed': ['Success', 'Failed'], 'type': 'string'}, 'Timestamp': {'maxlength': 250, 'minlength': 1, 'required': True, 'type': 'string',
'unique': True}}}
schema = {'App': {'maxlength': 15, 'minlength': 1, 'type': 'string'}, 'Info': {'maxlength': 300, 'minlength': 1, 'type': 'string'}, 'Status': {'allowed': ['Success', 'Failed'], 'type': 'string'}, 'Timestamp': {'maxlength': 250,
'minlength': 1, 'required': True, 'type': 'string', 'unique': True}}
```

Screenshot of log in database



```
ZacharySigamony@oz-ist-linux-fa18-411:~/abist411fa18Team2/redo/App3/App5$ python3 run.py
 * Serving Flask app "eve" (lazy loading)
 * Environment: production
   WARNING: Do not use the development server in a production environment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5010/ (Press CTRL+C to quit)
127.0.0.1 - - [11/Nov/2018 13:42:36] "POST /log HTTP/1.1" 201 -
127.0.0.1 - - [11/Nov/2018 13:42:36] "POST /log HTTP/1.1" 201 -
127.0.0.1 - - [11/Nov/2018 13:42:37] "POST /log HTTP/1.1" 201 -
127.0.0.1 - - [11/Nov/2018 13:43:11] "POST /log HTTP/1.1" 201 -
127.0.0.1 - - [11/Nov/2018 13:53:30] "POST /log HTTP/1.1" 201 -
127.0.0.1 - - [11/Nov/2018 13:53:30] "POST /log HTTP/1.1" 201 -
127.0.0.1 - - [11/Nov/2018 13:53:30] "POST /log HTTP/1.1" 201 -
127.0.0.1 - - [11/Nov/2018 13:53:31] "POST /log HTTP/1.1" 201 -
127.0.0.1 - - [11/Nov/2018 13:53:58] "POST /log HTTP/1.1" 201 -
127.0.0.1 - - [11/Nov/2018 13:53:58] "POST /log HTTP/1.1" 201 -
127.0.0.1 - - [11/Nov/2018 13:56:29] "POST /log HTTP/1.1" 201 -
127.0.0.1 - - [11/Nov/2018 13:56:29] "POST /log HTTP/1.1" 201 -
127.0.0.1 - - [11/Nov/2018 13:57:16] "POST /log HTTP/1.1" 201 -
127.0.0.1 - - [11/Nov/2018 14:16:15] "POST /log HTTP/1.1" 201 -
127.0.0.1 - - [11/Nov/2018 14:16:15] "POST /log HTTP/1.1" 201 -
127.0.0.1 - - [11/Nov/2018 14:16:26] "POST /log HTTP/1.1" 201 -
127.0.0.1 - - [11/Nov/2018 14:16:26] "POST /log HTTP/1.1" 201 -
127.0.0.1 - - [11/Nov/2018 14:30:17] "POST /log HTTP/1.1" 201 -
127.0.0.1 - - [11/Nov/2018 14:30:17] "POST /log HTTP/1.1" 201 -
127.0.0.1 - - [11/Nov/2018 14:30:18] "POST /log HTTP/1.1" 201 -
127.0.0.1 - - [11/Nov/2018 14:30:25] "POST /log HTTP/1.1" 201 -
127.0.0.1 - - [11/Nov/2018 14:30:25] "POST /log HTTP/1.1" 201 -
127.0.0.1 - - [11/Nov/2018 15:08:24] "POST /log HTTP/1.1" 201 -
127.0.0.1 - - [11/Nov/2018 15:08:24] "POST /log HTTP/1.1" 201 -
127.0.0.1 - - [11/Nov/2018 15:08:24] "POST /log HTTP/1.1" 201 -
127.0.0.1 - - [11/Nov/2018 15:08:56] "POST /log HTTP/1.1" 201 -
127.0.0.1 - - [11/Nov/2018 15:09:00] "POST /log HTTP/1.1" 201 -
127.0.0.1 - - [11/Nov/2018 15:09:00] "POST /log HTTP/1.1" 201 -
127.0.0.1 - - [11/Nov/2018 15:09:00] "POST /log HTTP/1.1" 201 -
127.0.0.1 - - [11/Nov/2018 15:09:00] "POST /log HTTP/1.1" 201 -
127.0.0.1 - - [11/Nov/2018 15:38:07] "POST /log HTTP/1.1" 201 -
127.0.0.1 - - [11/Nov/2018 15:38:07] "POST /log HTTP/1.1" 201 -
127.0.0.1 - - [11/Nov/2018 15:40:37] "POST /log HTTP/1.1" 201 -
127.0.0.1 - - [11/Nov/2018 15:40:37] "POST /log HTTP/1.1" 201 -
127.0.0.1 - - [11/Nov/2018 15:40:38] "POST /log HTTP/1.1" 201 -
127.0.0.1 - - [11/Nov/2018 15:40:51] "POST /log HTTP/1.1" 201 -
127.0.0.1 - - [11/Nov/2018 15:40:55] "POST /log HTTP/1.1" 201 -
127.0.0.1 - - [11/Nov/2018 15:40:55] "POST /log HTTP/1.1" 201 -
127.0.0.1 - - [11/Nov/2018 15:40:55] "POST /log HTTP/1.1" 201 -
127.0.0.1 - - [11/Nov/2018 15:40:55] "POST /log HTTP/1.1" 201 -
127.0.0.1 - - [11/Nov/2018 15:42:34] "POST /log HTTP/1.1" 201 -
127.0.0.1 - - [11/Nov/2018 15:42:34] "POST /log HTTP/1.1" 201 -
127.0.0.1 - - [11/Nov/2018 15:43:21] "POST /log HTTP/1.1" 201 -
127.0.0.1 - - [11/Nov/2018 16:05:39] "POST /log HTTP/1.1" 201 -
```

Screenshot of App5 running



RickyZhao@oz-ist-linux-fa18-411: ~/ProjectDiamond/abist411fa18Team2

```
{ "_id" : ObjectId("5be8460ce4804457602040b3"), "Timestamp" : "2018-11-11 15:09:00.682550", "App" : "App1", "Status" : "Success", "Info" : "Retrieved
 JSON payload from URL", "_updated" : ISODate("2018-11-11T15:09:00Z"), "_created" : ISODate("2018-11-11T15:09:00Z"), "_etag" : "67c448b61a8924ef54e17
b51959b95e93ea52ade" }
{ "_id" : ObjectId("5be8460ce4804457602040b4"), "Timestamp" : "2018-11-11 15:09:00.688707", "App" : "App1", "Status" : "Success", "Info" : "Saved JSO
N payload", "_updated" : ISODate("2018-11-11T15:09:00Z"), "_created" : ISODate("2018-11-11T15:09:00Z"), "_etag" : "4c953526834c41e07235985eeaf388510c
9737dd" }
{ "_id" : ObjectId("5be8460ce4804457602040b5"), "Timestamp" : "2018-11-11 15:09:00.705893", "App" : "App1", "Status" : "Success", "Info" : "Sending J
son Payload", "_updated" : ISODate("2018-11-11T15:09:00Z"), "_created" : ISODate("2018-11-11T15:09:00Z"), "_etag" : "4262bd989713bf63e09d498d1b25a8fa
ae350df9" }
{ "_id" : ObjectId("5be8460ce4804457602040b6"), "Timestamp" : "2018-11-11 15:09:00.710948", "App" : "App2", "Status" : "Success", "Info" : "Recieved
JSON payload", "_updated" : ISODate("2018-11-11T15:09:00Z"), "_created" : ISODate("2018-11-11T15:09:00Z"), "_etag" : "63fadbda453536305e09a00aee2e74b
492e4abb8" }
{ "_id" : ObjectId("5be84cdfe4804457602040b7"), "Timestamp" : "2018-11-11 15:38:07.555819", "App" : "App2", "Status" : "Success", "Info" : "Hashed JS
ON Payload", "_updated" : ISODate("2018-11-11T15:38:07Z"), "_created" : ISODate("2018-11-11T15:38:07Z"), "_etag" : "2d31a4efcf80a8426b6c69cbd19c48870
a0b2ca7" }
{ "_id" : ObjectId("5be84cdfe4804457602040b8"), "Timestamp" : "2018-11-11 15:38:07.565752", "App" : "App2", "Status" : "Success", "Info" : "Hashed JS
ON payload", "_updated" : ISODate("2018-11-11T15:38:07Z"), "_created" : ISODate("2018-11-11T15:38:07Z"), "_etag" : "7dff6801e1c6cd6f4f6644beca4fc89d6
31ba1a9" }
{ "_id" : ObjectId("5be84d75e4804457602040b9"), "Timestamp" : "2018-11-11 15:40:37.106430", "App" : "App3", "Status" : "Success", "Info" : "Opened JS
ON file", "_updated" : ISODate("2018-11-11T15:40:37Z"), "_created" : ISODate("2018-11-11T15:40:37Z"), "_etag" : "2f85d296a7036a46b72f9c7eb4060fed0da9
85fe" }
{ "_id" : ObjectId("5be84d75e4804457602040ba"), "Timestamp" : "2018-11-11 15:40:37.115651", "App" : "App3", "Status" : "Success", "Info" : "Hashed JS
ON data", "_updated" : ISODate("2018-11-11T15:40:37Z"), "_created" : ISODate("2018-11-11T15:40:37Z"), "_etag" : "8ea0bd7230c1fbc976f7d5b9444af75d91d6
e1f8" }
{ "_id" : ObjectId("5be84d76e4804457602040bb"), "Timestamp" : "2018-11-11 15:40:38.165801", "App" : "App3", "Status" : "Success", "Info" : "Sent JSON
 payload to email addresses", "_updated" : ISODate("2018-11-11T15:40:38Z"), "_created" : ISODate("2018-11-11T15:40:38Z"), "_etag" : "bae2fc3bf7608478
11cae8c5378249d61aae8590" }
{ "_id" : ObjectId("5be84d82e4804457602040bc"), "Timestamp" : "2018-11-11 15:40:50.995018", "App" : "App2", "Status" : "Success", "Info" : "Started s
erver", "_updated" : ISODate("2018-11-11T15:40:50Z"), "_created" : ISODate("2018-11-11T15:40:50Z"), "_etag" : "14bf3feb713363fa582d7e6a15ce60e6441019
4c" }
{ "_id" : ObjectId("5be84d87e4804457602040bd"), "Timestamp" : "2018-11-11 15:40:55.425276", "App" : "App1", "Status" : "Success", "Info" : "Retrieved
 JSON payload from URL", "_updated" : ISODate("2018-11-11T15:40:55Z"), "_created" : ISODate("2018-11-11T15:40:55Z"), "_etag" : "d4c34def3c9cecc3e0046
1cec5d7bfb96cc75c7f" }
{ "_id" : ObjectId("5be84d87e4804457602040be"), "Timestamp" : "2018-11-11 15:40:55.434991", "App" : "App1", "Status" : "Success", "Info" : "Sending J
son Payload", "_updated" : ISODate("2018-11-11T15:40:55Z"), "_created" : ISODate("2018-11-11T15:40:55Z"), "_etag" : "64f86c82ea554238508ade5d8168bd3b
519db319" }
{ "_id" : ObjectId("5be84d87e4804457602040bf"), "Timestamp" : "2018-11-11 15:40:55.440577", "App" : "App2", "Status" : "Success", "Info" : "Recieved
JSON payload", "_updated" : ISODate("2018-11-11T15:40:55Z"), "_created" : ISODate("2018-11-11T15:40:55Z"), "_etag" : "a09da52d845753b3bfce58247a9d99b
e6f56c82b" }
{ "_id" : ObjectId("5be84d87e4804457602040c0"), "Timestamp" : "2018-11-11 15:40:55.441389", "App" : "App1", "Status" : "Success", "Info" : "Saved JSO
N payload", "_updated" : ISODate("2018-11-11T15:40:55Z"), "_created" : ISODate("2018-11-11T15:40:55Z"), "_etag" : "f51f9b48743fe1c18d433f4ecd82d80253
a53e87" }
{ "_id" : ObjectId("5be84deae4804457602040c1"), "Timestamp" : "2018-11-11 15:42:34.386736", "App" : "App3", "Status" : "Success", "Info" : "Sent JSON
 payload to App4", "_updated" : ISODate("2018-11-11T15:42:34Z"), "_created" : ISODate("2018-11-11T15:42:34Z"), "_etag" : "9524a50c946d1ab196ec6514e87
b570af51e45d3" }
{ "_id" : ObjectId("5be84deae4804457602040c2"), "Timestamp" : "2018-11-11 15:42:34.394860", "App" : "App3", "Status" : "Success", "Info" : "Compresse
d JSON file", "_updated" : ISODate("2018-11-11T15:42:34Z"), "_created" : ISODate("2018-11-11T15:42:34Z"), "_etag" : "d672538ebaf11988367c32c60b2b6ab4
02f985fc" }
Type "it" for more
> it
{ "_id" : ObjectId("5be84e19e4804457602040c3"), "Timestamp" : "2018-11-11 15:43:21.525529", "App" : "App2", "Status" : "Success", "Info" : "Hashed JS
ON payload", "_updated" : ISODate("2018-11-11T15:43:21Z"), "_created" : ISODate("2018-11-11T15:43:21Z"), "_etag" : "7bd66b8e6abd189203b23ee23ee304a0c
41ca7e5" }
>
```

Screenshot of log in database