```python
from flask import Flask,jsonify,request,make_response
import subprocess
import json
import http
import itertools
import jwt
import datetime
import os
from functools import wraps

app = Flask(__name__)

cmdlist = ["ifconfig", "echo"] #Control list: enter any commands here
login = True
app.config['SECRET_KEY'] ='thisisthesecretkey'
authlog = open("authorizedlog.txt", "a+") #authorized log file
unauthlog = open("unauthorizedlog.txt", "a+") #unauthorized log file

def tokenRequired(f): #checks the token for the /bashcall URL
  @wraps(f)
  def decorated(*args, **kwargs):
    token = request.args.get('token')
    ipaddr = request.remote_addr
    if not token: #if token is wrong, record in unauthlog.txt
      unauthlog = open("unauthorizedlog.txt","a+")
      unauthsize = os.path.getsize("unauthorizedlog.txt")

      if unauthsize == 0: # If file is empty, add headers
        left_aligned = 'Timestamp:'
        center = 'Command/Token Put:'
        right_aligned = 'Source IP Address:'
        "{left_aligned:<15}{center:^10}{right_aligned:>15}".format(
          left_aligned = left_aligned,
          center = center,
          right_aligned = right_aligned)
        unauthlog.write(left_aligned.ljust(15))
        unauthlog.write(center.center(31))
        unauthlog.write(right_aligned.rjust(21))
        unauthlog.write("\n")

      dt = datetime.datetime.utcnow()
      unauthlog.write(dt.strftime("%m/%d/%Y %H:%M:%S")) #timestamp
      unauthlog.write("   ")
      unauthlog.write("NO TOK")
      unauthlog.write(ipaddr.rjust(30))
      unauthlog.write("\n")
      return jsonify({'message': "Unauthorized Request"}), 403

    try:
      data = jwt.decode(token, app.config['SECRET_KEY']) #decoding token

    Except: #any exception with token results in recording in unauthlog
      unauthlog = open("unauthorizedlog.txt","a+")
      unauthsize = os.path.getsize("unauthorizedlog.txt")

      if unauthsize == 0: #if file is new, add headers
        left_aligned = 'Timestamp:'
        center = 'Command/Token Put:'
        right_aligned = 'Source IP Address:'
        "{left_aligned:<15}{center:^10}{right_aligned:>15}".format(
          left_aligned = left_aligned,
          center = center,
          right_aligned = right_aligned)
        unauthlog.write(left_aligned.ljust(15))
        unauthlog.write(center.center(31))
        unauthlog.write(right_aligned.rjust(21))
        unauthlog.write("\n")

      dt = datetime.datetime.utcnow()
      unauthlog.write(dt.strftime("%m/%d/%Y %H:%M:%S"))
      unauthlog.write("   ")
      unauthlog.write("NO TOK")
      unauthlog.write(ipaddr.rjust(30))
      unauthlog.write("\n")
      return jsonify({'message': "Unauthorized Request"}), 403

    return f(*args, **kwargs)

  return decorated

@app.route("/login") #login route: returns token if password is correct
def login():
  auth = request.authorization

  if auth and auth.password == 'password': #can make ‘password’ whatever
    token = jwt.encode({'user': auth.username, 'exp' :
datetime.datetime.utcnow() + datetime.timedelta(minutes=30)},
app.config['SECRET_KEY']) #token expiry: 30 min after now
    return jsonify({'token' : token.decode('UTF-8')})
  return make_response('Could not verify!\n', 401, {'WWW-Authenticate' :
'Basic realm="Login Required"'}) #if login is not working


@app.route("/bashcall", methods = ["POST"]) #main URL, runs bash command

@tokenRequired #calls the tokenRequired() to deal with token validation

def runCommand(): #runs the command if command is in control list
  if login == False: #checking login functionality
    errlogin = "No login recognized. Please login using the '/login'
url\n"
    return errlogin

  str = request.json['command']
  str_in_list = any(map(str.__contains__, cmdlist))#if command in list
  ipaddr = request.remote_addr #ip address logger

  if str_in_list == False:
    errstr = "Operation not Permitted\n"
    dt = datetime.datetime.utcnow()
    unauthlog = open("unauthorizedlog.txt","a+")
    unauthsize = os.path.getsize("unauthorizedlog.txt")

    if unauthsize == 0: #if file is new, add headers
      left_aligned = 'Timestamp:'
      center = 'Command/Token Put:'
      right_aligned = 'Source IP Address:'
      "{left_aligned:<15}{center:^10}{right_aligned:>15}".format(
        left_aligned = left_aligned,
        center = center,
        right_aligned = right_aligned)
      unauthlog.write(left_aligned.ljust(15))
      unauthlog.write(center.center(31))
      unauthlog.write(right_aligned.rjust(21))
      unauthlog.write("\n")

    unauthlog.write(dt.strftime("%m/%d/%Y %H:%M:%S")) #timestamp
    unauthlog.write("   ")
    unauthlog.write(str) #command ran
    unauthlog.write(ipaddr.rjust(30))
    unauthlog.write("\n")
    return errstr

  dt = datetime.datetime.utcnow()#current time
  authlog = open("authorizedlog.txt", "a+")
  authsize = os.path.getsize("authorizedlog.txt")

  if authsize == 0: #if file empty, add headers
    left_aligned = 'Time Stamp:'
    center = 'Command Ran:'
    right_aligned = 'Source IP Address:'
    f"{left_aligned:<15}{center:^10}{right_aligned:>15}"

    authlog.write(left_aligned.ljust(15))
    authlog.write(center.center(25))
    authlog.write(right_aligned.rjust(25))
    authlog.write("\n")

  authlog.write(dt.strftime("%m/%d/%Y %H:%M:%S"))
  #authlog.write("\n")
  authlog.write("   ")
  authlog.write(str)
  authlog.write(ipaddr.rjust(23))
  authlog.write("\n")
  authlog.close()

  console = open("output.txt",'w+') #output file
  subprocess.call(str, shell=True, stdout=console) #runs the bash command
  console.close() #closes file

  text = open("output.txt",'r').read() #reads output cmd,returns to client
  return text



if __name__ == '__main__':
  app.run(port=5000) #hosts locally on port 5000
```