Technical Assessment – Code Review

Assume a coworker has just finished a task and has assigned a pull request to you for review. They were tasked to write some code to create an order through their order service. Your job is to identify and leave and in-line comments on any pieces of code that you identify to be problematic in terms logic, style, or anything else. For brevity, if there are multiple instances of a particular issue, feel free to leave a multi-line comment at the top of the module. You can assume the code is written using the latest release of Python 3 and the module that your coworker has exported will be called through and API request from the internet, with no prior processing done. Assume all required modules and their functions are tested and work. Please submit a plain text file containing the module below with your comments added. The file name should match the following: **firstname-lastname-pr.txt**

Time to complete: ~20 minutes

See below for code sample

```python
import db
import requests
from check_token_valid import check_token_valid
from make_db_order import make_db_order

'''
  payload : {
    name {
      first,
      last
    },
    token, cart_id, user_id
  }
'''

def create_order(payload):
    i = payload['cart_id']
    id = payload['user_id']
    first_name = payload['name']['first']
    last_name = payload['name']['last']
    user_token = payload['token']

    try:
        res = check_token_valid(user_token, id)
    except:
        return

    res = requests.get('http://myapi.com/cart' + i)

    valid_items = []

    for i in res.body.items:
        if i['id']:
            if i['price']:
                if i['name']:
                    valid_items.extend([i])
                else:
                    raise ValueError("Menu item name is  mandatory;")
            else:
                raise ValueError("Price is a required field")
        else:
            raise ValueError("Id is a required field")

    cursor = db.cursor()
    cursor.execute("select * from user where id=" + id)
    res = cursor.fetchall()
    res = chk_user(res.body)
    if res:
        order = make_db_order(valid_items)
        return order

# we should allow access to only to users of TYPE='app_user' and not
archived
def chk_user(user):
    if user.type == "ap_user":
        if user.archived == "true":
            raise KeyError("User is not active")
        return True
    return False
```