Students name :

Amr Jamal Eshtiwi 11840180 **Ahmed Abdallah Al-Qerem 11819195**

In this project we will build a micro service to implement an e-shop to sell books, this shop was implemented using three servers (front-end, catalog , order) built using **flask microservice** the first one shows the front-end “user interface” from this page the user can access the feature in the e-shop.

The features was :

1. /sreach → to show all books in database.
2. /search/<category> → to show all books that have same category.
3. /info/< item_number > → to show information for specific book.
4. /purchase/< item_number > → to purchase specific book from e-shop (through order server).

This URL will go to catalog server and get all the values and resend it to front-end server to show output at page browser this routing algorithm is applied on /sreach , /search/<category> and

/info/< item_number > the output is json and we implement every feature as bellow:

/sreach

The first thing the url will go to front-end server in my computer

```
@app.route("/")
@app.route("/home")
def home():
    return "BAZAR.COM ---- please write on URL what you need"

@app.route("/search", methods=['GET'])
def allBooks():
    return requests.get("http://192.168.1.17:5000/search").content
```

Then will go to catalog server that put in a virtual machine using http://192.168.1.17:5000/search and implement bellow code

```python
@app.route("/search", methods=['GET'])
def allBooks():
    file = open('catalog.csv')
    s = ""
    for line in csv.DictReader(file):
        line.pop('quantity')
        line.pop('price')
        s += json.dumps(line, indent=4)
    file.close()
    return s
```

Same thing for other features

/search/<category>

On my computer

```python
@app.route("/search/<category>", methods=['GET'])
def searchCatagory(category):
    url = "http://192.168.1.17:5000/search/"+category
    return requests.get(url).content
```

On virtual machine

```python
@app.route("/search/<category>", methods=['GET'])
def searchCatagory(category):
    file = open('catalog.csv')
    s = ""
    flag = 0
    for line in csv.DictReader(file):
        if line['Category'] == category:
            flag = 1
            line.pop('Category')
            line.pop('quantity')
            line.pop('price')
            s += json.dumps(line, indent=4)
    file.close()
    if flag == 0:
        s += "No matching category"
    return s
```

/info/<item_number>

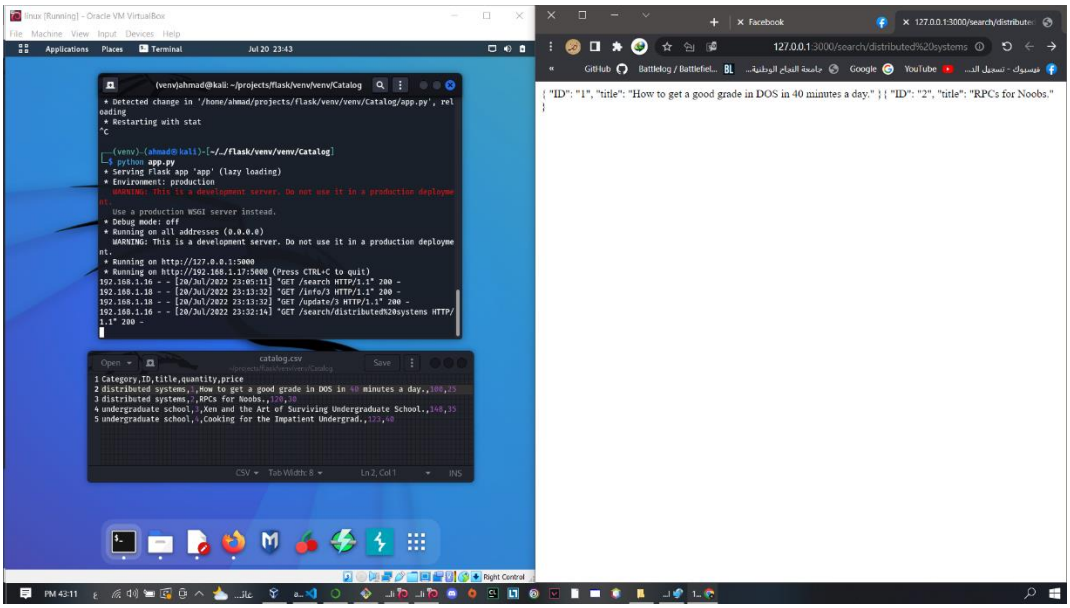
On my computer

```
@app.route("/info/<item_number>", methods=['GET'])
def bookInfo(item_number):
    url = "http://192.168.1.17:5000/info/"+item_number
    return requests.get(url).content
```

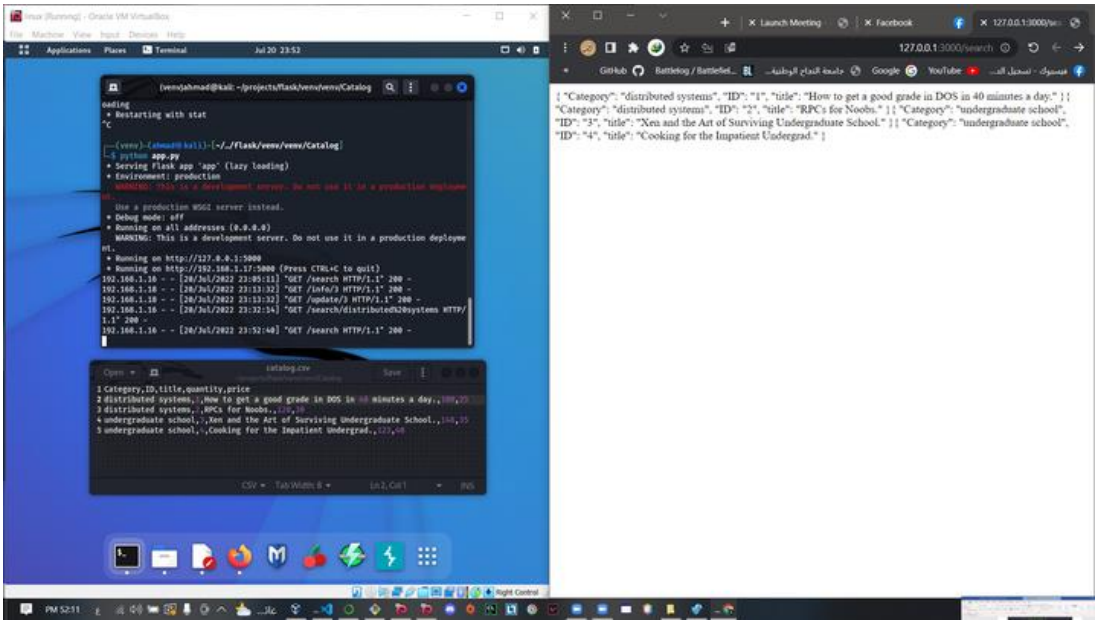
On virtual machine

```
@app.route("/info/<item_number>", methods=['GET'])
def bookInfo(item_number):
    file = open('catalog.csv')
    s = ""
    flag = 0
    for line in csv.DictReader(file):
        if line['ID'] == item_number:
            flag = 1
            line.pop('Category')
            line.pop('ID')
            s += json.dumps(line, indent=4)
    file.close()
    if flag == 0:
        s += "Item not found :("
    return s
```

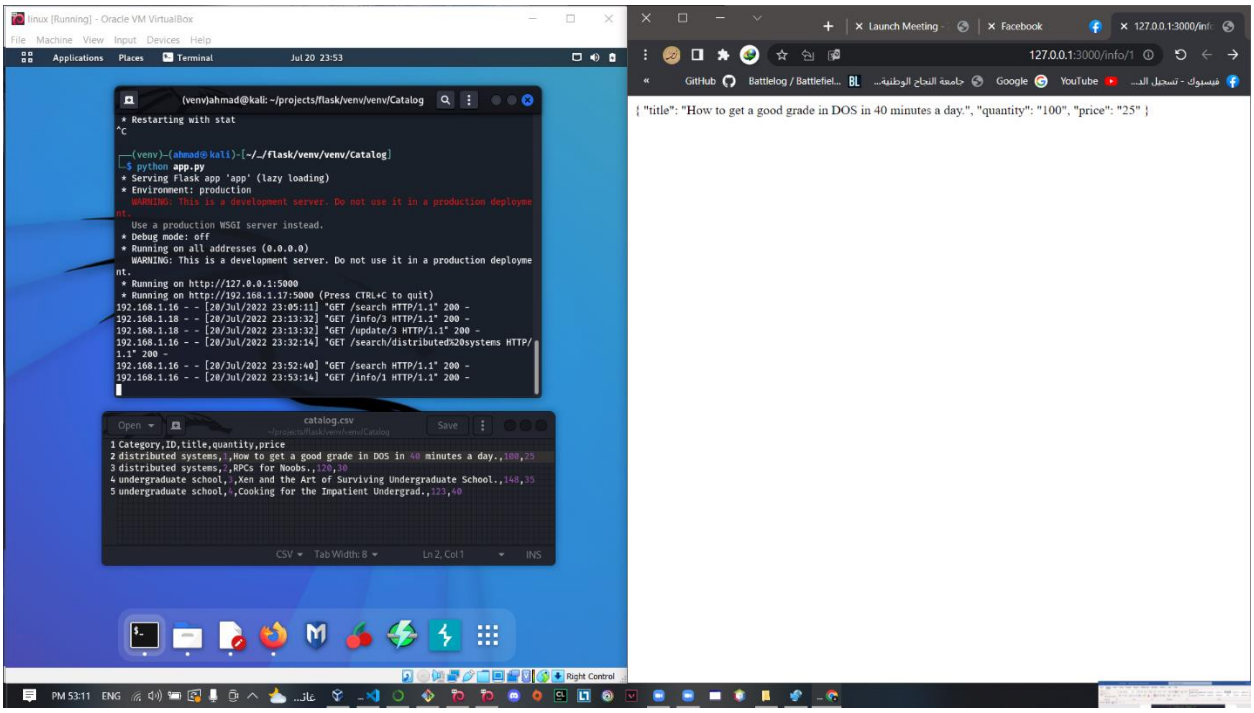
And the output for "/search/<category> "is :



And the output for "/search "is :



And the output for “info/< item_number > “is :



The last thing is the **purchase operation**
this operation is go through three servers

1- Front-end that routing to order server which is put on a virtual machine

```
@app.route("/purchase/<item_number>", methods=['GET'])
def purchase(item_number):
    url = "http://192.168.1.18:5000/purchase/"+item_number
    return requests.get(url).content
```

2- Order server which send request to catalog server to check if the book is still exist in the store or not
If exist it send another request to update the quantity of book to finish the purchase operation
And if not exist it sends to front-end that the book not found

```python
@app.route("/purchase/<item_number>", methods=['GET'])
def purchaseCatServer(item_number):
    # check quantity in stock
    url = "http://192.168.1.17:5000/info/"+item_number

    msg =requests.get(url)

    if msg.content.decode() == "Item not found :(":
        return msg.content
    quantity = int(msg.json()['quantity'])
    if quantity > 0:
        #if available in stock, update the quantity from the catalog server
        url = "http://192.168.1.17:5000/update/"+item_number
        bookName =requests.get(url).content

        s = "Purchase complete" + bookName.decode('UTF-8')
        return json.dumps(s), 200, {'ContentType': 'application/json'}
    else:
        #if not in stock, return failure message
        return json.dumps("Purchase failed"), 400, {'ContentType': 'application/json'}
```

3- Catalog server accepts update requests from order server

```python
@app.route("/update/<item_number>", methods=['GET'])
def update(item_number):
    df = pd.read_csv('catalog.csv')
    bookName =str(df.loc[int(item_number) - 1, 'title'])
    quantity =df.loc[int(item_number) - 1, 'quantity'] - 1
    df.loc[int(item_number)- 1, 'quantity'] = quantity
    df.to_csv('catalog.csv', index=False)
    return bookName
```

Notes:

-we use request library to send requests

-we use pandas to deal with CSV