

Python Programming

Lab Exercise- 3:

Submission Date:

5th August 2024

Submitted to:

Mrs. Sandhiya B.

Submitted by:

Eileen Maria Tom


(2447118)

SmartScan Codes:

This exercise creates a Python module named `smartscan_registration_module.py` that includes In-Memory Storage that creates a new user record, inserts and fetches all user records from the list. The next part of this exercise includes SmartScan Code Scanning where a function is implemented to read and decodes the QR code. The QR code shows the user information encoded as a comma-separated string. User Registration Function helps to implement a function `RegisterUserFromSmartScan` that extracts user data, uses the lambda functions to create and insert the user record into the in-memory list and prints the list of all registered users after adding the new user.

Code, Implementation and Output:

- (a) A .py file is created with the name “`smartscan_registration_module`” to define various functions. Here, as per specifications, lambda is used as a small anonymous function to create a concise code.

 jupyter `smartscan_registration_module.py`

An in-memory storage is created to store the data retrieved from the user. Then using lambda function, a new user record is created inserted and fetched from the list.

```
1 # a) In-Memory Storage
2 user_database=[]
3
4 # To create a new user record
5 create_new_user=lambda name, email: {"name": name, "email": email}
6
7 # To insert the user record into the list
8 insert_user=lambda user_record: user_database.append(user_record)
9
10 # To fetch all user records from the list
11 fetch_all_users=lambda: user_database
```

Here, `read_decode_smartscan()` is used to help with reading and decoding the SmartScan code and the data is stored in a comma-separated format. In case the data is in the wrong format, an error message is thrown.

```
13 # To create a function that reads and decodes the SmartScan Code
14 def read_decode_smartscan(code):
15     try:
16         name,email=code.split(',')
17         print(f"Name:{name}, Email:{email}")
18         return name,email
19     except ValueError:
20         print("Invalid Format! Please re-enter the values")
21         return None,None
```

The `RegisterUserFromSmartScan()` function uses the `read_decode_smartscan()` to extract the user data and uses lambda to create and insert a record in memory list.

```
# For user registration function
def RegisterUserFromSmartScan(code):
    name,email=read_decode_smartscan(code) # using the scanning function to extract user data
    if name and email:
        user_record = create_new_user(name, email) # using lambda function to create record in-memory list
        insert_user(user_record) # using lambda function to insert the record in-memory list
```

(b). A file called the “SmartScan_main.ipynb” is created to import and invoke various `def()` functions from the module `smartscan_registration_module`.

The pip install qrcode[pil] code helps to install the qrcode library along with the pillow library which is used for image processing.

```
[51]: pip install qrcode[pil]
```

```
Collecting qrcode[pil]
  Downloading qrcode-7.4.2-py3-none-any.whl.metadata (17 kB)
Requirement already satisfied: typing-extensions in c:\users\hp\anaconda3\lib\site-packages (from qrcode[pil]) (4.11.0)
Requirement already satisfied: pypng in c:\users\hp\anaconda3\lib\site-packages (from qrcode[pil]) (0.20220715.0)
Requirement already satisfied: colorama in c:\users\hp\anaconda3\lib\site-packages (from qrcode[pil]) (0.4.6)
Requirement already satisfied: pillow>=9.1.0 in c:\users\hp\anaconda3\lib\site-packages (from qrcode[pil]) (10.3.0)
Downloading qrcode-7.4.2-py3-none-any.whl (46 kB)
----- 0.0/46.2 kB ? eta -:--:--
----- 41.0/46.2 kB 960.0 kB/s eta 0:00:01
----- 46.2/46.2 kB 765.0 kB/s eta 0:00:00
Installing collected packages: qrcode
Successfully installed qrcode-7.4.2
Note: you may need to restart the kernel to use updated packages.
```

Then by importing the newly created module and entering the data, that is name and email of three different persons. And by using print(), the data is displayed.

```
[2]: import smartscan_registration_module
ss_code=["AllenParker,allenp@gmail.com", "KaranKumar,karankrai@outlook.com", "TinaJoe,tina12@gmail.com"]
```

```
[4]: print(ss_code)

['AllenParker,allenp@gmail.com', 'KaranKumar,karankrai@outlook.com', 'TinaJoe,tina12@gmail.com']
```

This code is used to fetches and prints the name and email of all the users entered in a list format.

```
[6]: # Using the RegisterUserFromSmartScan function
for code in ss_code:
    smartscan_registration_module.RegisterUserFromSmartScan(code)
print("The list of all registered users:\n",smartscan_registration_module.fetch_all_users())

Name:AllenParker, Email:allenp@gmail.com
Name:KaranKumar, Email:karankrai@outlook.com
Name:TinaJoe, Email:tina12@gmail.com
The list of all registered users:
[{'name': 'AllenParker', 'email': 'allenp@gmail.com'}, {'name': 'KaranKumar', 'email': 'karankrai@outlook.com'}, {'name': 'TinaJoe', 'email': 'tina12@gmail.com'}]
```

These set of codes are responsible for generating a QR code, creating an image file for the QR code, checking whether the directory to save the QR code exists or not and saving the QR code in .png file.

```
•[28]: # To import the necessary libraries for QR code generation
import qrcode
import os

•[44]: def generate_qr_code(name, email, filename):
    data = f"{name},{email}"

    # To generate the QR code
    qr = qrcode.QRCode(
        version=1,
        error_correction=qrcode.constants.ERROR_CORRECT_L,
        box_size=10,
        border=4,
    )
    qr.add_data(data)
    qr.make(fit=True)

    img = qr.make_image(fill='black', back_color='white') # To create an image file for QR code
    img.save(filename)
    print(f"QR code saved as {filename}")

def generate_qr_codes_for_users(users, directory='qr_codes'):
    os.makedirs(directory, exist_ok=True) # to check if the directory exists

    for user in users:
        name, email = user['name'], user['email']
        filename = os.path.join(directory, f"{name.replace(' ', '_')}.png")
        generate_qr_code(name, email, filename)

# All user data to add 3 different QR codes
users = [{"name": "Allen Parker", "email": "allenp@gmail.com"},
        {"name": "Karan Kumar", "email": "karankrai@outlook.com"},
        {"name": "Tina Joe", "email": "tina12@gmail.com"}]

dir_path = 'C:\\Users\\hp\\Documents\\qr_codes' # To save the QR code to local file
generate_qr_codes_for_users(users, dir_path)

QR code saved as C:\Users\hp\Documents\qr_codes\Allen_Parker.png
QR code saved as C:\Users\hp\Documents\qr_codes\Karan_Kumar.png
QR code saved as C:\Users\hp\Documents\qr_codes\Tina_Joe.png
```

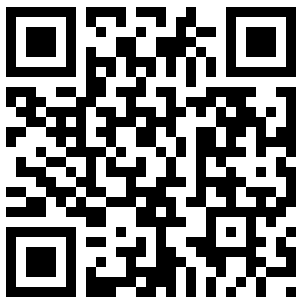
The QR code is customized with a box size of 10 and border length of 4 with the image file for QR code. Then the os library help to check if the directory exists or not and the directory path or the path of the local file where the QR code can be saved is mentioned. Finally, three separate QR code is generated for three different data.

The QR code generated for all three data is given below:

Allen Parker-



Karan Kumar-



Tina Joe-

