New Devices Lab Report
First Draft

Andrei Spiridon; s1027022

December 2021

1 Introduction

My project is called "the smart-closet". It's main functionality is allowing the user to store their clothes' attributes in a local database through a bar-code scanner; once their clothes are registered the system will be able to suggest outfits for the user based on the weather outside and their scheduled events for the day.

2 Design

My original goal was to make a smart mirror which would have the functionality mentioned above embedded in its system. I realized quickly that making a smart mirror would over complicate the project and go past the desired goal. As such, I limited myself to implementing only the outfit suggesting feature as well as the clothes registration feature.

I have chosen to use a bar code scanner out of all the "registration processes" as it provides a simple and cheap interface to uniquely identify each clothing item. It will be connected to a Raspberry Pi, as it provides all of the functionality I need, such as storing a clothes database and running Python scripts.

With the completed product, the user will be able to scan a clothing item; if it is a new clothing item the user will be prompted to input it's characteristics (e.g.: Shoe, Casual, Black, Textile, Hip, etc.), otherwise it's value in the column "in wardrobe" will be negated. This way, the system will be able to keep track of which clothes are present in the wardrobe and not suggest dirty clothes or ones the user uses no more.

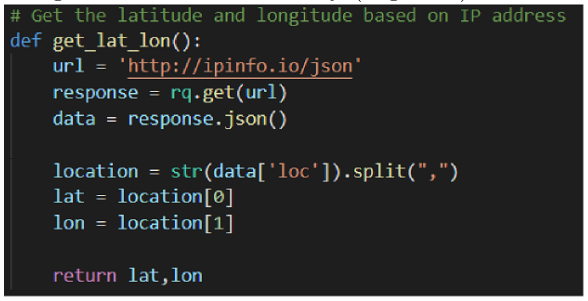
When the main script runs, the user will be presented with the current weather, their scheduled events for the day and the style chosen by the system; they will also be given the option to change the style to one of their own choosing. Once this process is completed, the system will suggest an outfit. The user will be given the opportunity to ask for another suggestion or change out specific clothing items.

3 Software design

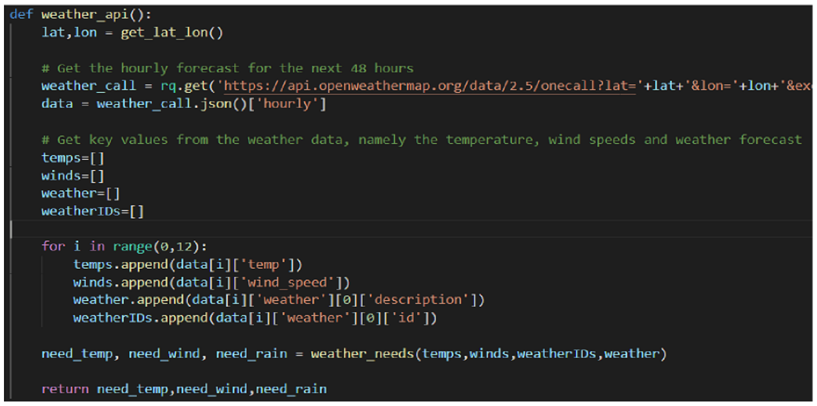
The main software operation is done through Python scripts.

3.1 Weather API

First Python script deals with getting the weather information from the **One Call API**. This is done through a simple API call including the user's latitude and longitude and their API key (Figure 1).



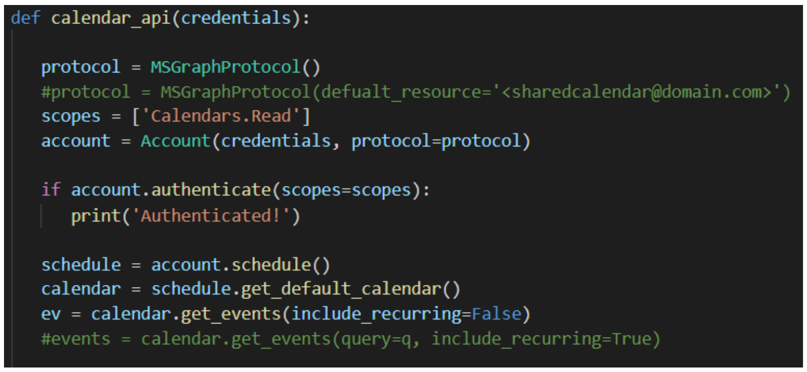
```
# Get the latitude and longitude based on IP address
def get_lat_lon():
    url = 'http://ipinfo.io/json'
    response = rq.get(url)
    data = response.json()

    location = str(data['loc']).split(",")
    lat = location[0]
    lon = location[1]

    return lat,lon
```



```
def weather_api():
    lat,lon = get_lat_lon()

    # Get the hourly forecast for the next 48 hours
    weather_call = rq.get('https://api.openweathermap.org/data/2.5/onecall?lat='+lat+'&lon='+lon+'&ex
    data = weather_call.json()['hourly']

    # Get key values from the weather data, namely the temperature, wind speeds and weather forecast
    temps=[]
    winds=[]
    weather=[]
    weatherIDs=[]

    for i in range(0,12):
        temps.append(data[i]['temp'])
        winds.append(data[i]['wind_speed'])
        weather.append(data[i]['weather'][0]['description'])
        weatherIDs.append(data[i]['weather'][0]['id'])

    need_temp, need_wind, need_rain = weather_needs(temps,winds,weatherIDs,weather)

    return need_temp,need_wind,need_rain
```

3.2 Calendar API

This script deals with getting the events from the user's Outlook calendar. This process is not very streamlined yet, as in order to call Microsoft's API, one needs to authenticate through OAuth2 by clicking a link and then pasting back into the console the resulting URL.

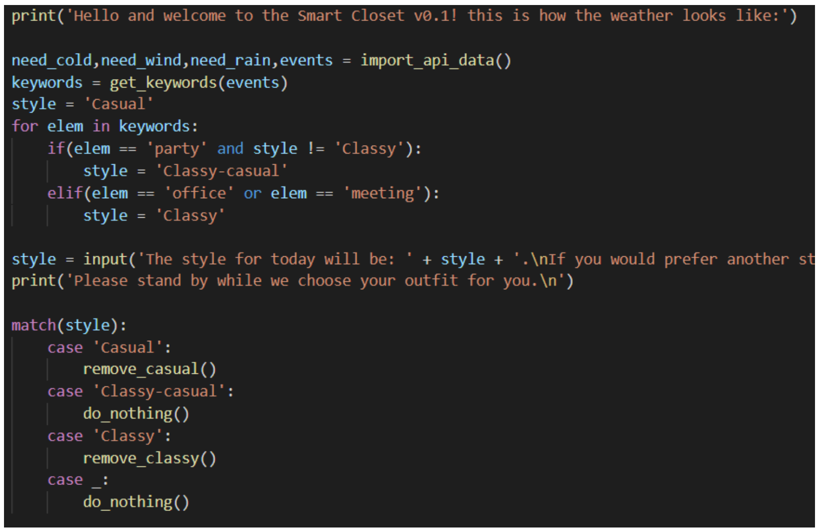

```
def calendar_api(credentials):

    protocol = MSGraphProtocol()
    #protocol = MSGraphProtocol(defualt_resource='<sharedcalendar@domain.com>')
    scopes = ['Calendars.Read']
    account = Account(credentials, protocol=protocol)

    if account.authenticate(scopes=scopes):
        print('Authenticated!')

    schedule = account.schedule()
    calendar = schedule.get_default_calendar()
    ev = calendar.get_events(include_recurring=False)
    #events = calendar.get_events(query=q, include_recurring=True)
```

3.3 Outfit chooser

This script deals with the literal outfit choosing operation. So far no progress has been made besides creating an `Outfit` class, so there is not much to show for.

3.4 Main script

As the name suggests, this is the script which makes direct use of all the other scripts. It contains a `main` method, where the flow of the algorithm is found.


```
print('Hello and welcome to the Smart Closet v0.1! this is how the weather looks like:')

need_cold,need_wind,need_rain,events = import_api_data()
keywords = get_keywords(events)
style = 'Casual'
for elem in keywords:
    if(elem == 'party' and style != 'Classy'):
        style = 'Classy-casual'
    elif(elem == 'office' or elem == 'meeting'):
        style = 'Classy'

style = input('The style for today will be: ' + style + '.\nIf you would prefer another st
print('Please stand by while we choose your outfit for you.\n')

match(style):
    case 'Casual':
        remove_casual()
    case 'Classy-casual':
        do_nothing()
    case 'Classy':
        remove_classy()
    case _:
        do_nothing()
```

Here, the style is chosen based on keywords parsed from the title of the events received from the calendar API. The user is then given the option of choosing another style. Afterwards, the full database is trimmed down to fit the chosen style. Here I used the new feature of Python 3.10, match case.

4 Usage guide

In this section I will discuss the steps needed to take in order to operate the smart closet. Since I have not yet tackled all parts of the project such as working with the bar-code scanner and Raspberry Pi, only the software side of the operations will be explored.

4.1 Software operation

The main script is run. The user is welcomed and presented with a link for the purpose of authenticating to the Microsoft Graph API. This will open a browser window, where the user has to log in with a Microsoft account. Afterwards, the resulting link is to be copied back into the console. After being presented with the system-chosen style, the user is given the option to enter another style. There is nothing more implemented so far.

5 Conclusion

As the project is not close to completion, I cannot say that I have reached my goals yet. What I can say is that I have made good progress given the fact that I work by myself and that I am sure to complete it.