## 5.2 Screen Shots of System Interfaces

This section displays the graphical user interface of the predicting missed medical appointment for *Pusat Rawatan Warga* UMS using machine learning system. Python is the language used to build the interface with the use of Tkinter, which is a Python binding to the TK GUI toolkit.

The main user interface of machine learning for *Pusat Rawatan Warga* UMS is shown in Figure 5.24. There are two buttons, the Login and Sign Up buttons. If the user is using this system for the first time, he or she needs to create an account by clicking the "Sign Up" button. However, if the user already has an account, he or she can press the 'Login' button directly.

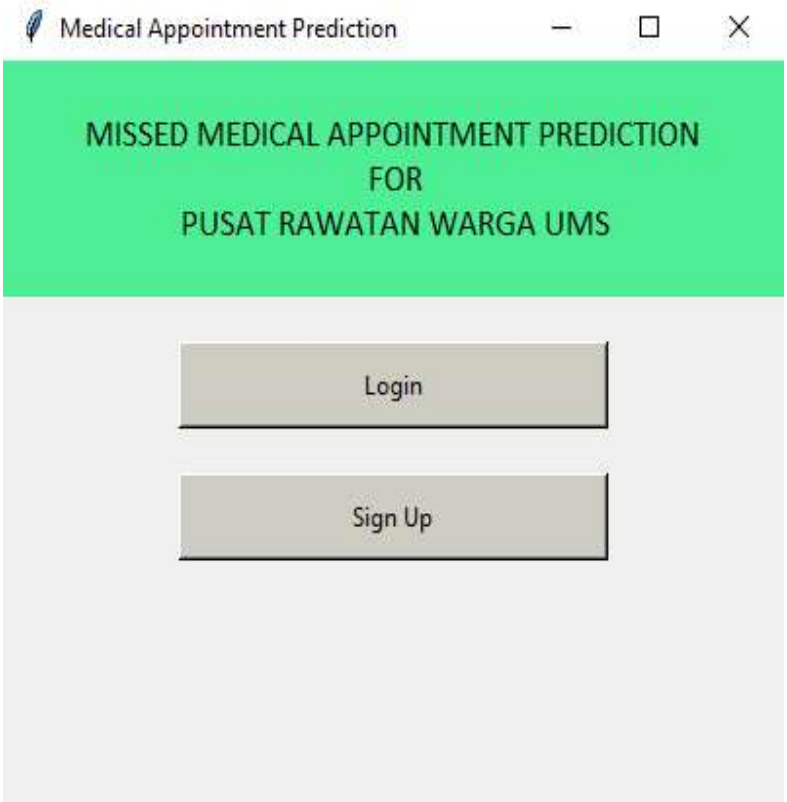



**Figure 5.24: Main User interface**

Figure 5.25 show the interfaces that will be displayed when the "Sign Up" button was clicked. Users will be asked to insert details in creating their account, such as their username and password. Then, the user must press the "Register" button to make the system save their account information. The registration notification that will notify the user that his or her registration has been successful is shown in Figure 5.26 below.

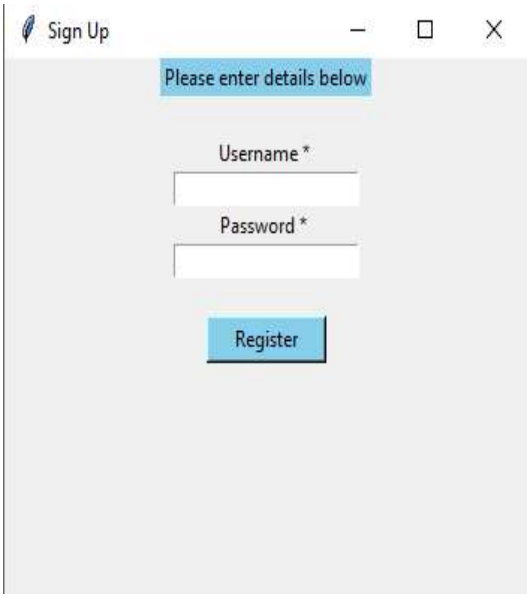



**Figure 5.25 : Sign Up Interface**

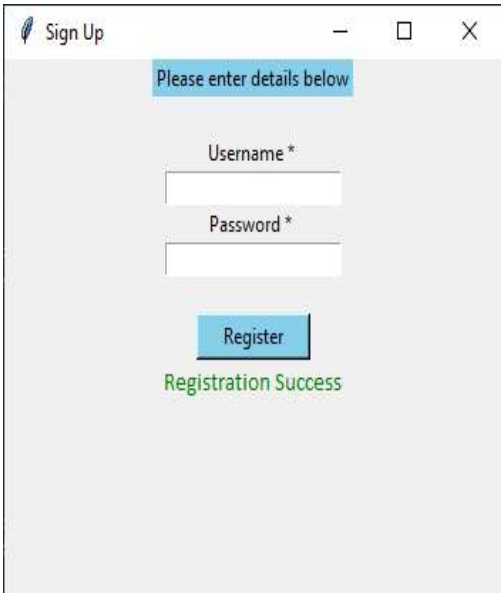



**Figure 5.26 : Success Registration Alert**

Figure 5.27 below display the login interface of the system. User need to input their username and password that they have registered in the system.

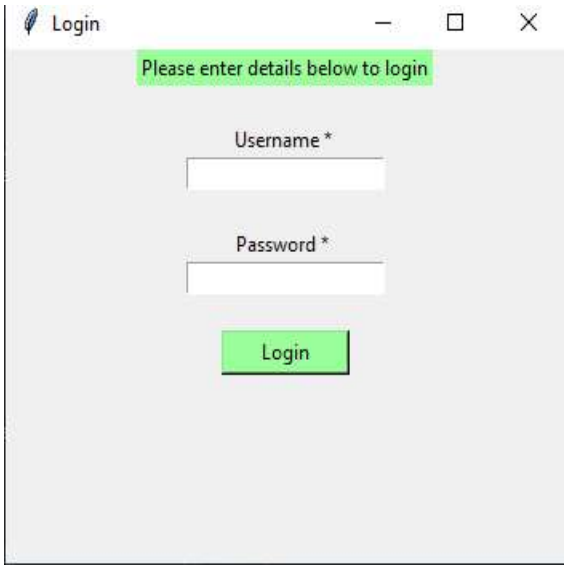
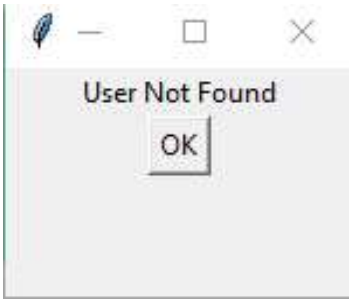



**Figure 5.27 : Login interface**

Figure 5.28 shows the user not found alert that will be displayed if the username and password that are not registered in the system yet has been entered or wrong character of username and password has been inserted. However, the login success alert that will be shown if the username and password have been entered correctly is shown in Figure 5.29.



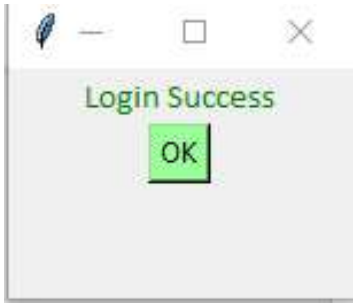




**Figure 5.28 : User Not Found Alert**

**Figure 5.29 : Login Success Alert**

Figure 5.30 presents the interface of the patient information store. When the user logs into the system successfully, after clicking on the "ok" button in the login success alert, they will see this interface directly. As what we can see in the Figure 5.31, the character switches to a blue colour after the patient information has been entered and the "save" button has been pressed, which means that the data entered has been saved in the csv file.

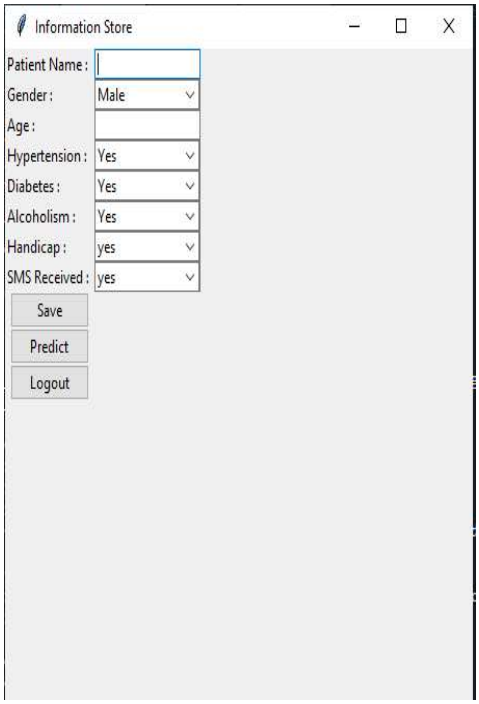



**Figure 5.30: Patient Information Saved Interface**

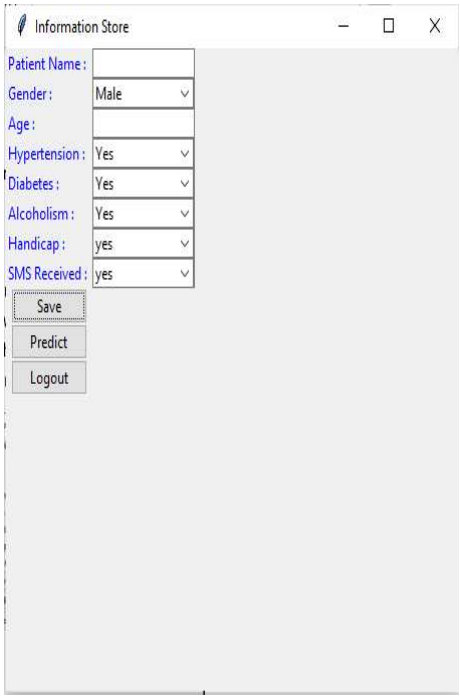



**Figure 5.31: After Data Store Interface**

Figure 5.32 and Figure 5.33 below display the result interface that indicates that the patient is coming for the appointment or not.

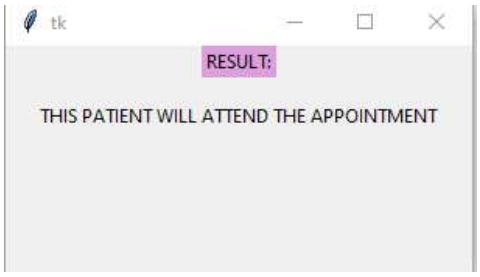



**Figure 5.32: Result Interface for Patient That Attend**

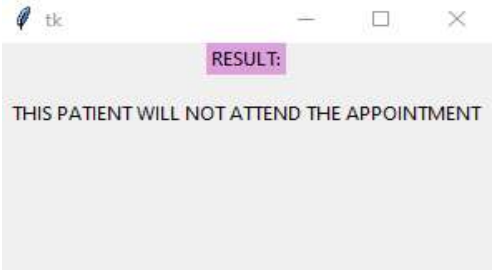



**Figure 5.33: Result Interface for Patient That Absent**

The logout interface that will appear if the "logout" button has been clicked on in the patient information store interface is shown in Figure 5.34 below. The "Yes" button will make the user interface disappear, while the "Cancel" button will make the user interface still appear.

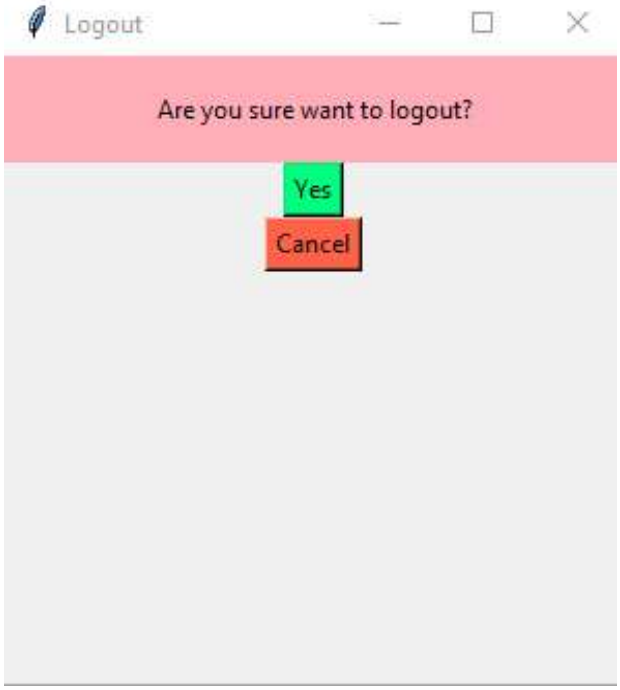


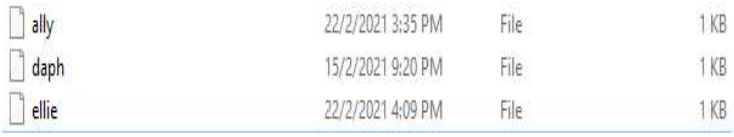

**Figure 5.34: Logout Interface**

## 5.3 Data Storage

The txt file and the csv file are the data storage which was used to build this machine learning system. Since the prediction made by the support vector machine learning built in this project was integrated with the data in the csv file, it is important for the user to insert some patient information to be the predictor in making prediction. In the user interface, all the information required by the system is used as the predictor to create the accurate prediction.

Firstly, user need to login into the system. Then, the patient data that the system needs should be inserted. After that, the data will be stored in the csv file to be the predictor for predicting whether a patient will come or not for the appointment. When this predictor is inserted into the csv file, the data can be processed by machine learning to generate the prediction result. Therefore, the machine learning system will give the output weather the patient will attend or not for the appointment, and this result will be shown in the user interface.

As what we can see in the Figure 5.35 below, the username and password of user that have been registered in the system will be directly stored in the txt file, so that the system may identified the registered user or not.



| | | | |
|---|---|---|---|
| ally | 22/2/2021 3:35 PM | File | 1 KB |
| daph | 15/2/2021 9:20 PM | File | 1 KB |
| ellie | 22/2/2021 4:09 PM | File | 1 KB |

**Figure 5.35: The file that save the registered user**

Figure 5.36 below shows the contain of the txt file when it was opened.


**Figure 5.36: The look of saved data in the txt file**

## 5.4 IDE Platform

The support vector machine learning complete software, including the graphical user interface coding was run in Spyder IDE.

The figures below display the codes that were used in developing this project. The three files connected with each other with their own functions.

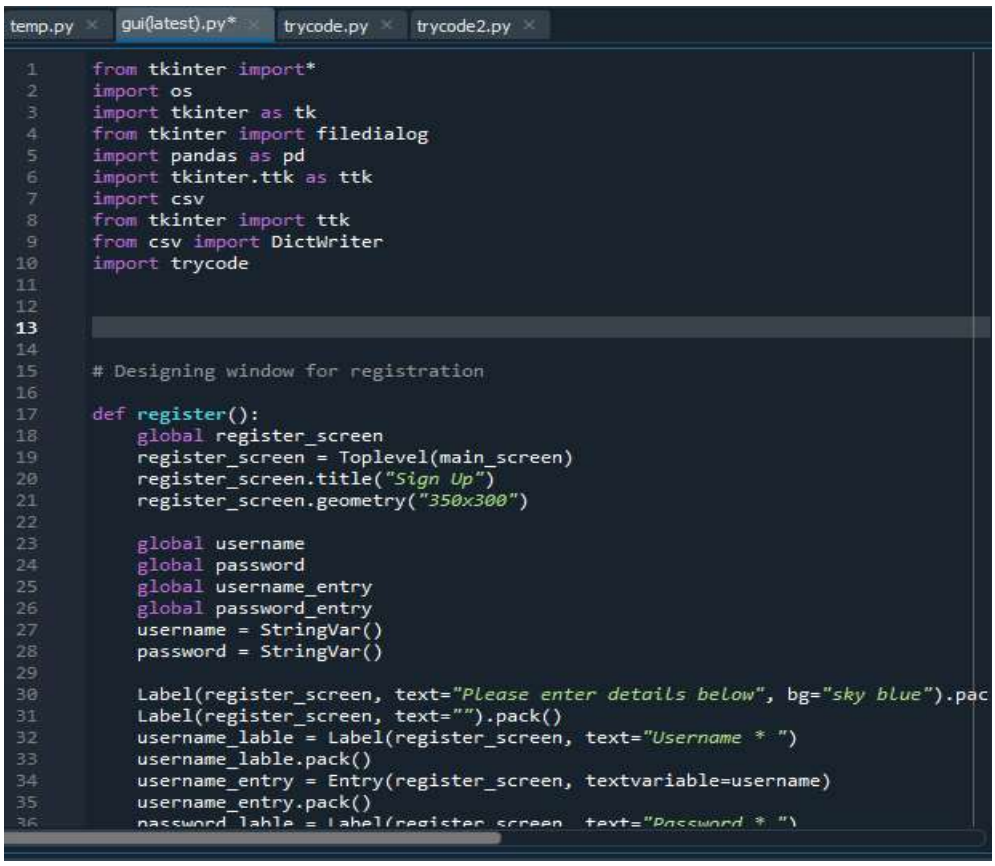



**Figure 5.37: The looks of code in the Spyder IDE**

temp.py | gui(latest).py | trycode.py | trycode2.py | trycode1.py

```python
import tkinter as tk    #import tkinter
from tkinter import ttk
from csv import DictWriter
import os
import trycode2
import pandas as pd #pandas is used to load & manipulate data and for one-Hot en
from tkinter import Tk, Label, Frame, Entry, Button
from tkinter import*
import trycode2

def insert_data():
    win = tk.Tk()
    win.geometry("450x450")
    win.title('Information Store')

    #create labels
    #patientid label
    patientid_label = ttk.Label(win,text = "Patient Name : ")
    patientid_label.grid(row=0, column=0, sticky = tk.W)

    #gender label
    gender_label = ttk.Label(win,text = "Gender : ")
    gender_label.grid(row=1, column = 0,sticky =tk.W)

    #age label
    age_label = ttk.Label(win,text = "Age : ")
    age_label.grid(row=2, column = 0, sticky = tk.W)

    #hypertension label
    hypertension_label = ttk.Label(win, text = "Hypertension : ")
    hypertension_label.grid(row=3, column = 0, sticky =tk.W)

    #diabetes label
    diabetes_label = ttk.Label(win,text = "Diabetes : ")
    diabetes_label.grid(row=4, column = 0, sticky = tk.W)
```

**Figure 5.38: The looks of code in the Spyder IDE**

temp.py | gui(latest).py | trycode.py | trycode2.py

```python
import pandas as pd #pandas is used to load & manipulate data and for one-Hot enc
import numpy as np #data manipulation
import matplotlib.pyplot as plt #to draw graphs
import matplotlib.colors as colors
from sklearn.model_selection import train_test_split #split data into training &
from sklearn.preprocessing import scale #scale and center data
from sklearn.svm import SVC #make s SVM for classification
from sklearn.model_selection import GridSearchCV #cross validation
from sklearn.metrics import confusion_matrix #create confusion matrix
from sklearn.metrics import plot_confusion_matrix #draws confusion matrix
from sklearn.decomposition import PCA #perform PCA to plot the data
from tkinter import*


#import the data
df = pd.read_csv('file.csv',
                 header=0) #The second line contains column name, so skip the firs

df.head() #Loaded data into data frame

df.dtypes #identify missing data

#To make sure it only contain certain number
df['Patient ID'].unique()

#To make sure it only contain certain number
df['Gender'].unique()

#To make sure it only contain certain number
df['Age'].unique()

#To make sure it only contain certain number
df['Hypertension'].unique()

#To make sure it only contain certain number
df['Diabetes'].unique()
```

**Figure 5.39: The looks of code in the Spyder IDE**