

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.

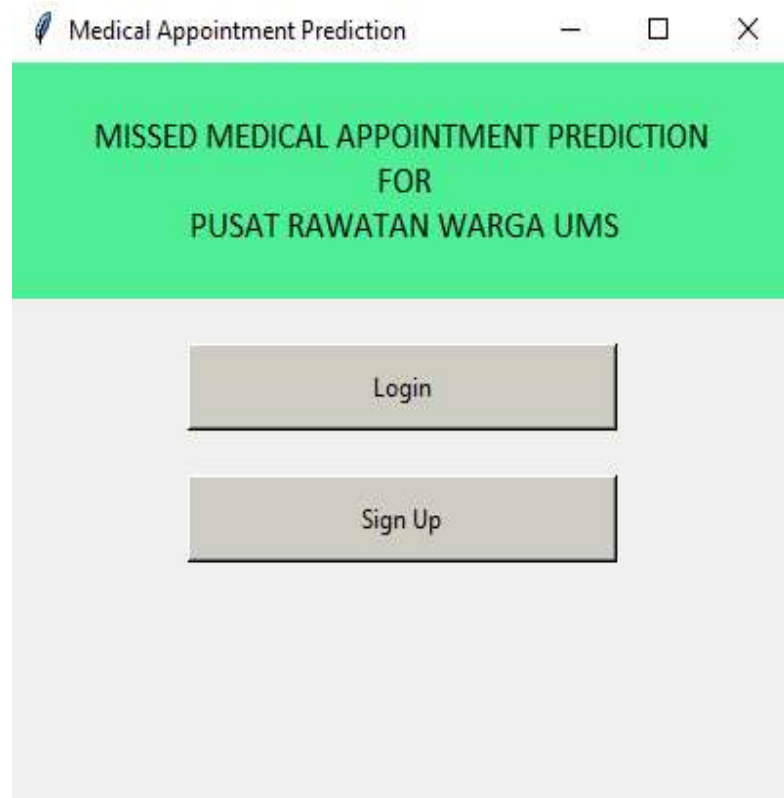


Figure 5.24: Main User interface

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.

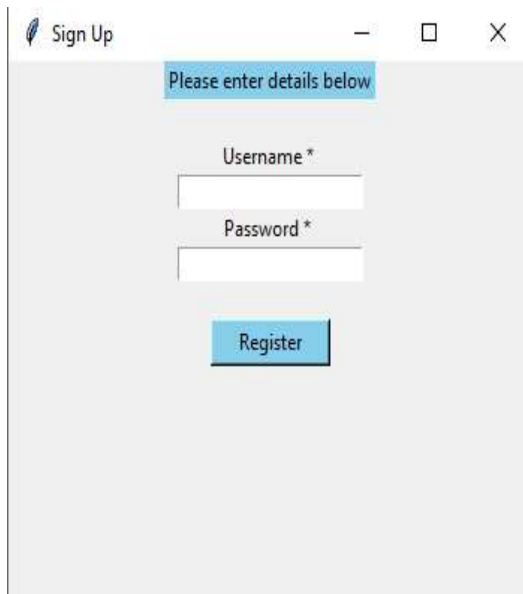
A screenshot of a web application window titled "Sign Up". The window has a light gray background and a blue header bar. Below the header, there is a blue button labeled "Please enter details below". Underneath this button, there are two input fields: "Username *" and "Password *". Below the input fields, there is a blue button labeled "Register".

Figure 5.25 : Sign Up Interface

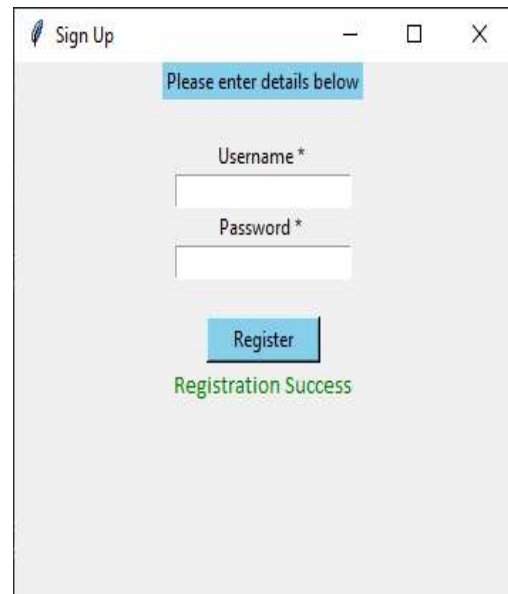
A screenshot of a web application window titled "Sign Up". The window has a light gray background and a blue header bar. Below the header, there is a blue button labeled "Please enter details below". Underneath this button, there are two input fields: "Username *" and "Password *". Below the input fields, there is a blue button labeled "Register". Below the "Register" button, there is a green text label "Registration Success".

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.

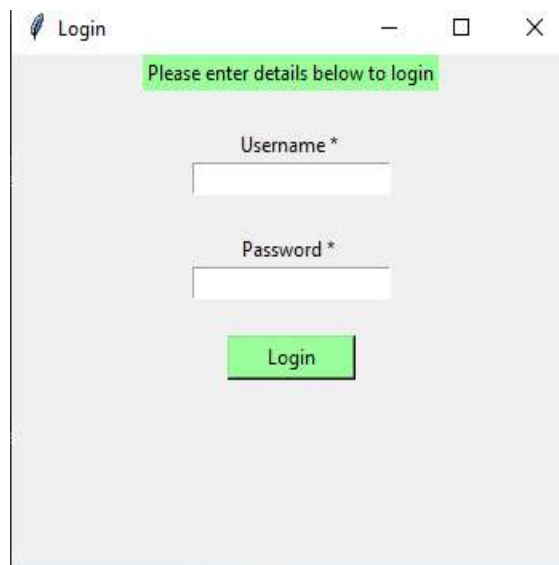
A screenshot of a login window titled "Login". At the top, there is a green message box that says "Please enter details below to login". Below this, there are two input fields: "Username *" and "Password *". Underneath the password field is a green "Login" button. The window has standard Windows-style controls (minimize, maximize, close) in the top right corner.

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.

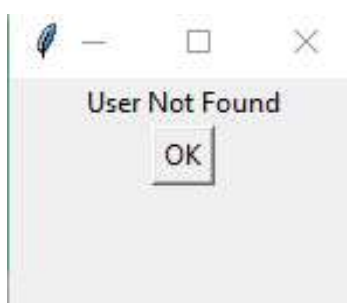


Figure 5.28 : User Not Found Alert



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.

Information Store

Patient Name:

Gender: Male ▾

Age:

Hypertension: Yes ▾

Diabetes: Yes ▾

Alcoholism: Yes ▾

Handicap: yes ▾

SMS Received: yes ▾

Save

Predict

Logout

Figure 5.30: Patient Information Saved Interface

Information Store

Patient Name:

Gender: Male ▾

Age:

Hypertension: Yes ▾

Diabetes: Yes ▾

Alcoholism: Yes ▾

Handicap: yes ▾

SMS Received: yes ▾

Save

Predict

Logout

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.

tk

RESULT:

THIS PATIENT WILL ATTEND THE APPOINTMENT

Figure 5.32: Result Interface for Patient That Attend

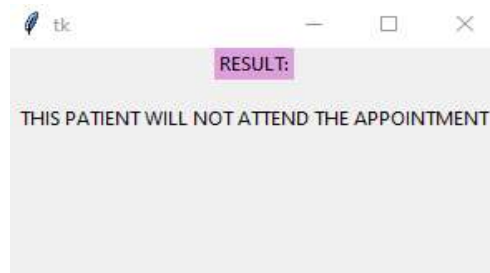


Figure 5.33: Result Interface for Patient That Absent

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.

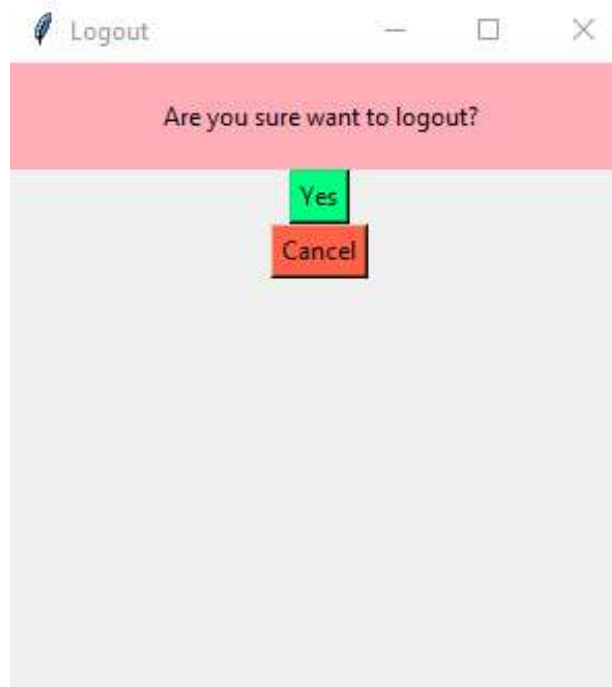



Figure 5.34: Logout Interface

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 whether 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.

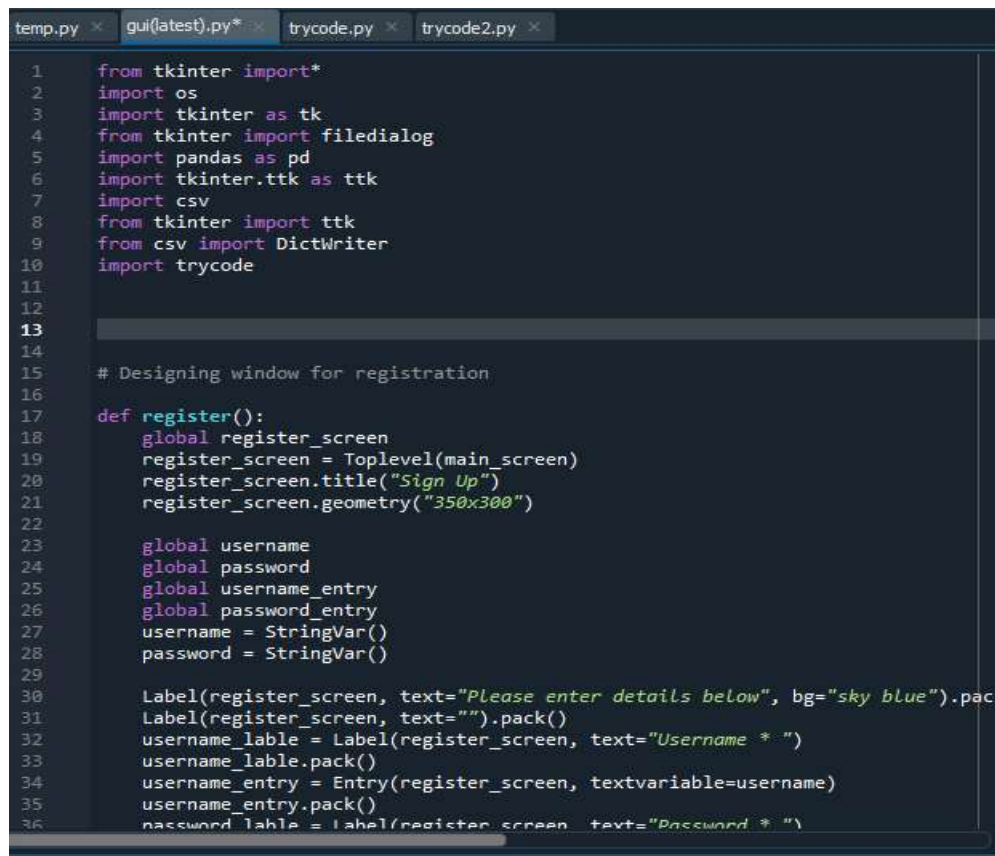


Figure 5.36: The look of saved data in the txt file

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.

A screenshot of the Spyder IDE interface. The top bar shows four open files: temp.py, gui(latest).py*, trycode.py, and trycode2.py. The main editor window displays the code for gui(latest).py. The code is a Python script for a registration window using Tkinter. It includes imports for tkinter, os, tkinter as tk, tkinter filedialog, pandas as pd, tkinter.ttk as ttk, csv, and trycode. The script defines a register() function that creates a Toplevel window titled "Sign Up" with a geometry of 350x300. It uses global variables for username and password, and creates labels and entry fields for the registration form. The code is as follows:

```
1  from tkinter import*
2  import os
3  import tkinter as tk
4  from tkinter import filedialog
5  import pandas as pd
6  import tkinter.ttk as ttk
7  import csv
8  from tkinter import ttk
9  from csv import DictWriter
10 import trycode
11
12
13
14
15 # Designing window for registration
16
17 def register():
18     global register_screen
19     register_screen = Toplevel(main_screen)
20     register_screen.title("Sign Up")
21     register_screen.geometry("350x300")
22
23     global username
24     global password
25     global username_entry
26     global password_entry
27     username = StringVar()
28     password = StringVar()
29
30     Label(register_screen, text="Please enter details below", bg="sky blue").pac
31     Label(register_screen, text="").pack()
32     username_label = Label(register_screen, text="Username * ")
33     username_label.pack()
34     username_entry = Entry(register_screen, textvariable=username)
35     username_entry.pack()
36     password_label = Label(register_screen, text="Password * ")
```

Figure 5.37: The looks of code in the Spyder IDE

```

1  import tkinter as tk #import tkinter
2  from tkinter import ttk
3  from csv import DictWriter
4  import os
5  import trycode2
6  import pandas as pd #pandas is used to load & manipulate data and for one-hot en
7  from tkinter import Tk, Label, Frame, Entry, Button
8  from tkinter import*
9  import trycode2
10
11 def insert_data():
12     win = tk.Tk()
13     win.geometry("450x450")
14     win.title('Information Store')
15
16     #create labels
17     #patientid label
18     patientid_label = ttk.Label(win, text = "Patient Name : ")
19     patientid_label.grid(row=0, column=0, sticky = tk.W)
20
21     #gender label
22     gender_label = ttk.Label(win, text = "Gender : ")
23     gender_label.grid(row=1, column = 0, sticky =tk.W)
24
25     #age label
26     age_label = ttk.Label(win, text = "Age : ")
27     age_label.grid(row=2, column = 0, sticky = tk.W)
28
29     #hypertension label
30     hypertension_label = ttk.Label(win, text = "Hypertension : ")
31     hypertension_label.grid(row=3, column = 0, sticky =tk.W)
32
33     #diabetes label
34     diabetes_label = ttk.Label(win, text = "Diabetes : ")
35     diabetes_label.grid(row=4, column = 0, sticky = tk.W)
36

```

Figure 5.38: The looks of code in the Spyder IDE

```

1  import pandas as pd #pandas is used to load & manipulate data and for one-hot enc
2  import numpy as np #data manipulation
3  import matplotlib.pyplot as plt #to draw graphs
4  import matplotlib.colors as colors
5  from sklearn.model_selection import train_test_split #split data into training &
6  from sklearn.preprocessing import scale #scale and center data
7  from sklearn.svm import SVC #make s SVM for classification
8  from sklearn.model_selection import GridSearchCV #cross validation
9  from sklearn.metrics import confusion_matrix #create confusion matrix
10 from sklearn.metrics import plot_confusion_matrix #draws confusion matrix
11 from sklearn.decomposition import PCA #perform PCA to plot the data
12 from tkinter import*
13
14
15 #import the data
16 df = pd.read_csv('file.csv',
17                 header=0) #The second line contains column name, so skip the first
18
19 df.head() #Loaded data into data frame
20
21 df.dtypes #identify missing data
22
23 #To make sure it only contain certain number
24 df['Patient ID'].unique()
25
26 #To make sure it only contain certain number
27 df['Gender'].unique()
28
29 #To make sure it only contain certain number
30 df['Age'].unique()
31
32 #To make sure it only contain certain number
33 df['Hypertension'].unique()
34
35 #To make sure it only contain certain number
36 df['Diabetes'].unique()
37

```

Figure 5.39: The looks of code in the Spyder IDE