PYTHON

JACKFRUIT PROBLEM

GROUP 4

- MAMIDI NAGA NIHARIKA **PES2UG25AM331**
- BHOOMIKA **PES2UG25CS636**
- GRACEMARY K ARALE **PES2UG25AM325**
- GOURI **PES2UG25CS645**

CLASS- SEM 1 / SECTION I

COLLEGE NAME - PES UNIVERSITY

ACADEMIC YEAR - 2025-2029

ACKNOWLEDGEMENT

We,the members of the project ‘HEALTHCARE’,would like to extend our deepest gratitude to everyone who contributed to the success of this project. This journey has been one of collaboration,learning,and mutual support,culminating in a project that we are proud to present.

First and foremost,we express our sincere thanks to our mentor, Prof.Ankita Singhai maam, for their valuable guidance and support in the preparation of this project.

We would also like to acknowledge the support of our families and friends,who provided encouragement,understanding and motivation throughout the duration of this project.

Lastly, we extend our gratitude to each other, the members of project ‘HEALTHCARE’

PROJECT HEALTHCARE

By- GROUP 4

SUBMITTED TO - ANKITA MAM

INDEX

INTRODUCTION

In this mini project, we have developed a simple health care system using python. The main aim of this project is to help users check basic health information and get useful suggestions. Python is an easy and powerful programming language. So it allows us to write programs that are user friendly and work smoothly.

This project takes basic inputs from the user like symptoms, height,weight,age or other details and processes them to give meaningful output. It also helps beginners understand how to use function,conditions,loops and file handling in real life applications.

This project is helpful because it shows how python can be used in the medical field for simple health-related tasks. It can help users get quick guidance, understand their health status and make basic decisions. Even though it is not a replacement for a doctor, it provides useful ideas like BMI calculation. Symptom checking and maintaining simple records.

HOW DOES IT WORK?

BMI CALCULATOR:

The BMI calculator asks the user to enter their height(in m) and weight(in kg).Using these values,the program calculates BMI using the formula:
BMI=weight/(height)^2

The program also determines the BMI category using if-else logic. The categories used are:

Underweight:BMI<18.5

Normal weight: 18.5 to 24.9

Overweight: 25.0 to 29.9

Obese: BMi>30

Symptom Checker:

The project uses basic if-else conditions to compare the users symptoms with commonly known diseases and display most likely condition. It is created for educational purposes and not for real medical diagnosis. The first step is to take input of the symptoms from the user .

The program compares the symptom with a list of predefined conditions.For each symptom ,a specific if-elif block is written. Each condition returns a possible disease. If the user enters a symptom not included in the code,the system returns a default message. It demonstrates how basic if else logic can be used to build a simple symptom checker in python.

Age group Analyzer(from fake patients data):

It analyzes the age of multiple patients and categorizes them into different age groups. The fake patients data can be stored in the form of datasets,csv files etc.The datasets are stored in a python list.

Each patient's age can be read and classified into different categories such as child,teenager,adult or senior citizen.The classification can be done using if-else logic.A for loop goes through each patient in the dataset,analyzes the age and prints the category.

Appointment Generator:

The appointment token generator takes the patient's name as input ,creates a unique 4-digit token number using python's random module,and records the current date and time using the datetime module. It then combines all this information and saves the token to a file -either a text file(.txt) or a pdf file(.pdf).It also allows the user to select their desired date on the calendar for his/her appointment.

FEATURES

1. import module's

1.import wx - GUI framework (wxPython)

I used the wxPython library to design the graphical user interface of the project. This module provides all the basic window components such as frames,panels,buttons,text boxes,and layout tools.

By importing wx, I was able to create the main application window,handle user inputs,and manage events like button clicks.

2.import re - regular expression module

The re module was used for validating user input. Whenever the program required checking formats(like email,numbers,specific patterns),regular expression helped to verify the correctness.

Using re made the data validation part more accurate and reliable.

3.import random - random number generation

The random module was used whenever the program needed random values,such as generating IDs,OTPs,or selecting random items.

This module gives functions like randint() and choice() which make randomization simple and effective.

4.import datatime - data and time handling

The project required handling dates and time in various places.

The datetime module was used to record the current date/time, calculate time differences,and format date values.

It helped in generating time stamps and managing any date-related operations.

5.import os - operating system utilities

The os module was used to interact with the file system.

It helped in tasks like checking whether a file exists,creating folders,listing directory contents,and reading/writing files.

Using os ensured that the program could work smoothly with the system’s file structure.

6.from pathlib import path -Object-Oriented file paths

The path class from the pathlib module made file and directory handling easier.instead of using normal strings for file paths,path provides clean and object-oriented methods for creating,joining,and checking file paths.

This improved the readability and maintainability of the code.

7.import wx.adv - Advanced wxPython Controls

Along with the basic wx widgets,the project also used some advanced GUI components like date pickers and notification tools.

The wx.adv module provides these additional features.

It was mainly used for adding controls like for datePickerctrl to allow users to choose dates in an easy and convenient way.

2. OOPS

We used this oops concept to perform inheritance of classes using a hierarchy model. We introduced child classes of a predefined wx.Dialog which is our parent class.

Oops concept deals with the integrating the classes and executing reusable functionality

It secures the data used and it is beneficial for data in bulk.

Here are the list of classes we used

```
class SignupDialog(wx.Dialog)

class SymptomDialog(wx.Dialog)

class AgeDialog(wx.Dialog)

class AppointmentDialog(wx.Dialog)

class MainFrame(wx.Frame)

class LoginFrame(wx.Frame)
```

3. def()

It is used to define a function block separately and even useful for executing it multiple times .

Here we use

```
def save_user(username, password, age, bloodgroup)

def check_credentials(username, password)

def append_appointment(name, token, dt, doc)

def on_signup(self, event)#Singup

def calculate(self, event)#BMI calculation

def on_check(self, event)#Check username and
password

def on_analyze(self, event)#Age analysing

def on_req(self, event)#Specification of doctors

def on_generate(self, event)#Token generating
```

Definitions a part of Login and Main frames-

```
def open_bmi(self, event)

def open_symptom(self, event)

def open_age(self, event)

def open_appointment(self, event)

def on_logout(self, event)

def on_login(self, event)

def on_signup(self, event)
```

4. wx features

1. wx.Frame - This feature is used in our program to create a gui based model which involves the visualisation of the screen in a structures format creating the frames. The main frame is the one which is carrying the signup and user login details.

2. wx.Panel - It enables users to create sections under a frame subdividing the frame for specific categories.

3. wx.App(False)- This helps creating an application and the ‘False’ helps executing the program within the GUI model itself.

4. wx.StaticText - It is used for the visibility of the data that does not change(static) .

5. wx.TextCtrl - It is used for creating text boxes so that the user can enter his details. We used it for entering his details like password,username etc… .

6. wx.Button - As the name suggests it creates buttons for functionality to proceed to further process just by one single click.

7. Button.bind - As mentioned previously this is the feature that lets your button click to move into a functionality via definitions(refer def()) to process further requirements.

8. wx.MessageBox - These are the pop up message boxes that indicate several mistakes,actions that are success or a failure. These can be represented in different signs including error, information etc… .

9. wx.Dialog - It is used to create pop up windows. These are followed up after specific buttons so the user has good clarity on what action he is currently performing.

10. wx.BoxSizer - It helps adjusting the size of the panel according to the compatibility of use.

11. wx.Choice - It helps create a drop down list which we used for selecting doctor's name and specification. Where the user selects predefined options.

12. app.MainLoop- It helps to let the application stand in until the user finishes their work with it.

13. wx.adv.DatePickerCtrl - This is a feature used in advance wx python. We used it for implementing a calendar module . Where the user can select the date of appointment.

5. Class features

1.Class - A class is a blueprint used to create objects in Python.It groups variables (data) and functions (methods) together.Classes help organize code and make it reusable.

2.def __init__(self,parent) - This is a constructor method that runs automatically when an object is created.It is used to initialize variables and set up the GUI layout.The parent parameter refers to the parent window or frame

3.self.EndModal()

This method ends the execution of a modal dialog.It allows the program to continue after the dialog is closed.Usually called when the user clicks OK or Cancel.

4.dlg.ShowModal() - Displays a dialog box in modal mode.The user must interact with this dialog before returning to the main window.

Program execution pauses until the dialog is closed.

5.dlg.Destroy() - Completely removes the dialog window from memory.It is used after the dialog is closed to free system resources.Helps prevent memory leaks in GUI applications.

6.self.Close()- Closes the current window or application frame.Any cleanup or close events linked to the window are triggered.The window is destroyed after closing.

7.self.Centre()- Positions the window at the center of the screen or parent window.Improves the appearance of the GUI.
Makes the window easier for users to see.

8.self.dictionary.items()-Returns all key-value pairs from a dictionary as tuples.Commonly used in loops to access both keys and values.
Helps process dictionary data efficiently.

9.self.choice.Clear()- Removes all existing items from a choice (dropdown) control.Used when the list needs to be refreshed.Prevents old data from remaining in the control.

10.self.choice.AppendItems()- Adds multiple items to a choice control at one time.
Makes updating dropdown lists faster than adding items individually.
Used after clearing old items.

11.self.choice.SetSelection(0)- Select the first item in the choice list.
Ensures there is always a default selected value.
Improves user interaction and avoids empty selection errors.

12.self.date_picker.GetValue()- Retrieves the date selected by the user from the date picker control.
Returns the date value in a usable format.
Used for validation, display, or saving data.

13.self.time_txt.GetValue()-Gets the text entered by the user in a time input field.
Returns the value as a string.
Used to process or validate time input.

14.self.Hide()- Hides the window without deleting it from memory.
The window can be shown again when needed.
Useful when switching between multiple windows or dialogs.

6. Additional features

Re module- a). re.compile()

This feature is used in our code while creating password, re.compile() prepares the pattern once.

Files- a).With open(file path),b) f.write() these features are used in opening the USER_FILE which has the data of username,password,age,blood group in write mode.And it automatically closes.

wx module- a) wx.EXPAND|wx.ALL is used for every button and textboxes to align them uniformly in a panel.

b) wx.ALIGN_CENTER|wx.ALL it aligns at the middle of the panel.

Datetime module-This module is used in appointment generator, it handles date and time properly.Date comes from calendar,time from text box , combine() merges them into one datetime.

Random module- random.randit(). This module generates random token number, used as appointment token.

Path module- a. Path.home()- extracts the address of the file from users folder in linux,mac etc. eg- names and niha file.

b. APP_DIR.mkdir()- creates the folder/directory if not present, prevents crash if folder exists

7. if -else- elif

These commands are the basics of python which help in choice making with the required condition.

It is also necessary that all the cases are included. There is how we use elif (if not the other condition(else-if)), and else when all the conditions are false.

8. Data types used

Majorly we require 2 datatypes

1.Lists[]- which store non homogeneous data which is enclosed in square brackets

2.Dictionary{}- This is a datatype which allows the user to store data in the form of key value pairs.

3.Tuples()-They are immutable data collections which store multiple items in a single row.

We used nested dictionaries like-

1. SYM_DICT = {1: ("Viral fever", "Note your temperature. If temperature between 99F to 103F - HIGH RISK."),

        2: ("Common cold", "Consult the doctor when the cold exceeds 7 days."),

        3: ("Food poisoning", "Consult doctor immediately!"),

        4: ("Common flu", "Consult the doctor when the cough exceeds 7 days."),

        5: ("Gastric problem", "For mild pain use Milk of magnesia."),

        6: ("Fungal skin infection", "Consult your dermatologist."),

        7: ("Cholera", "Consult doctor immediately!"),

        8: ("Migraine", "Don't use medications without doctor's consult."),

        9: ("Possible heart problem", "Get scans and consult cardiologist."),

```
        10:("Vertigo", "Get your appointment
ready.")}
```

2.
   ```
   DOCTORS = {1: ("General Physician",
   ["Dr.Anil","Dr.Jaya"]),
           2: ("Cardiologist",
   ["Dr.Ram","Dr.Yajas"]),
           3: ("Pediatrician",
   ["Dr.Arvind","Dr.Sudhanva"]),
           4: ("Dermatologist",
   ["Dr.Vartika","Dr.Aditi"]),
           5: ("Psychiatrist",
   ["Dr.Krishna","Dr.Yamini"])}
   ```

9. for loop and while loop

These loops function in a systematic format otherwise iterate until the end statement they help in working with the same logic for different values of the data and work continuously until we mention the end point

10. Basic statements

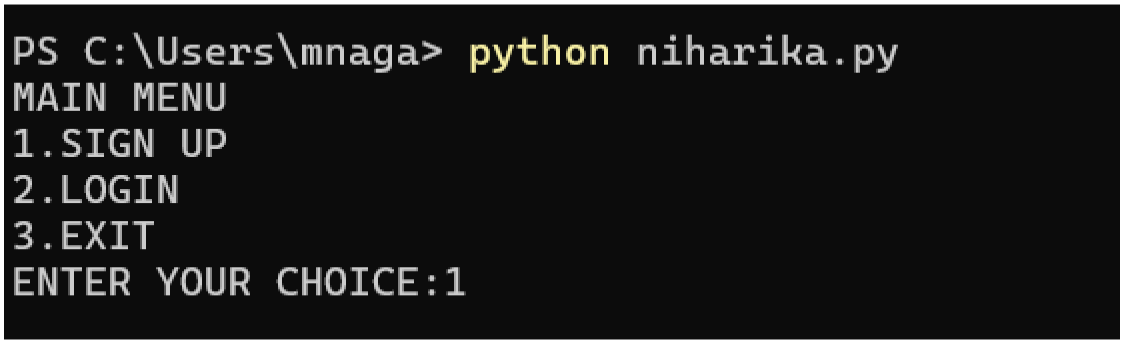
print() - it is a command useful to print the exact statement in output and also used in displaying the required data.

input()- it is a key feature used in python in order to take required input from the user and later allows it to be processed for later use.

OUTPUT

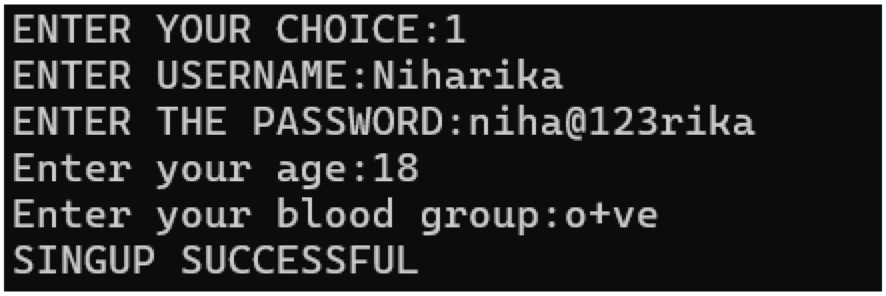
TERMINAL

1.main()



```
PS C:\Users\mnaga> python niharika.py
MAIN MENU
1.SIGN UP
2.LOGIN
3.EXIT
ENTER YOUR CHOICE:1
```

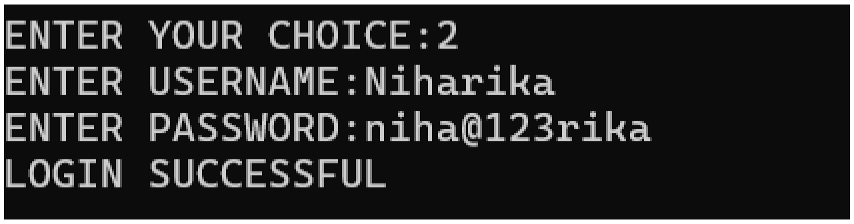
2.signup()



```
ENTER YOUR CHOICE:1
ENTER USERNAME:Niharika
ENTER THE PASSWORD:niha@123rika
Enter your age:18
Enter your blood group:o+ve
SINGUP SUCCESSFUL
```

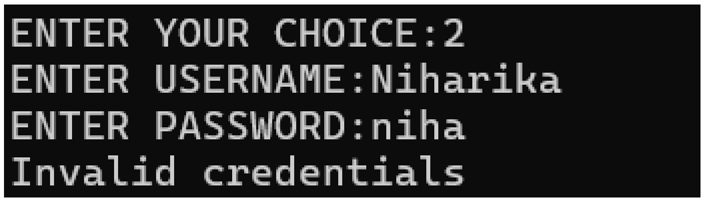
3.login()

SUCSEESFUL



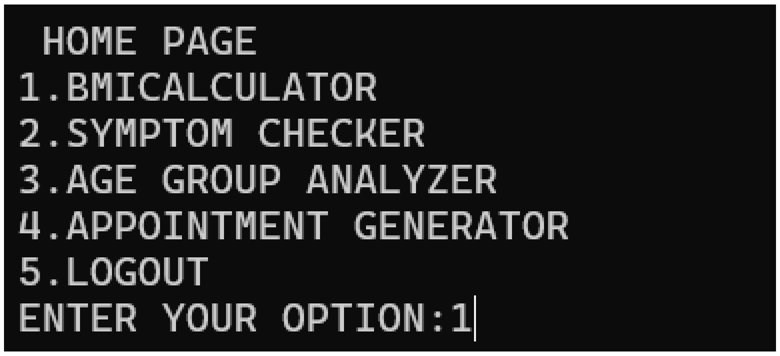
```
ENTER YOUR CHOICE:2
ENTER USERNAME:Niharika
ENTER PASSWORD:niha@123rika
LOGIN SUCCESSFUL
```

FAIL



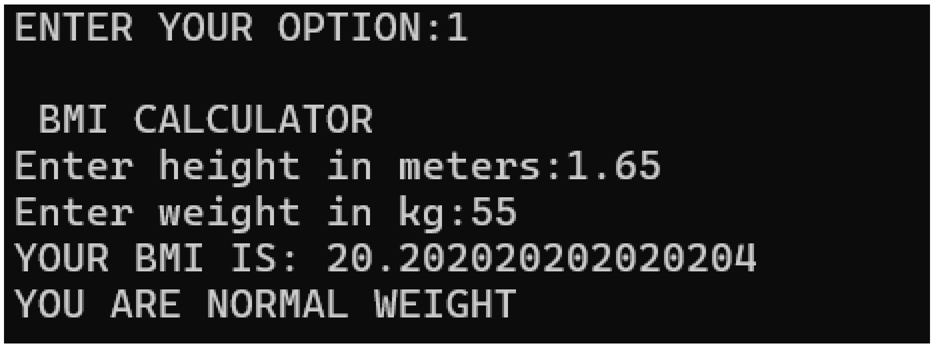
```
ENTER YOUR CHOICE:2
ENTER USERNAME:Niharika
ENTER PASSWORD:niha
Invalid credentials
```

4.homepage()



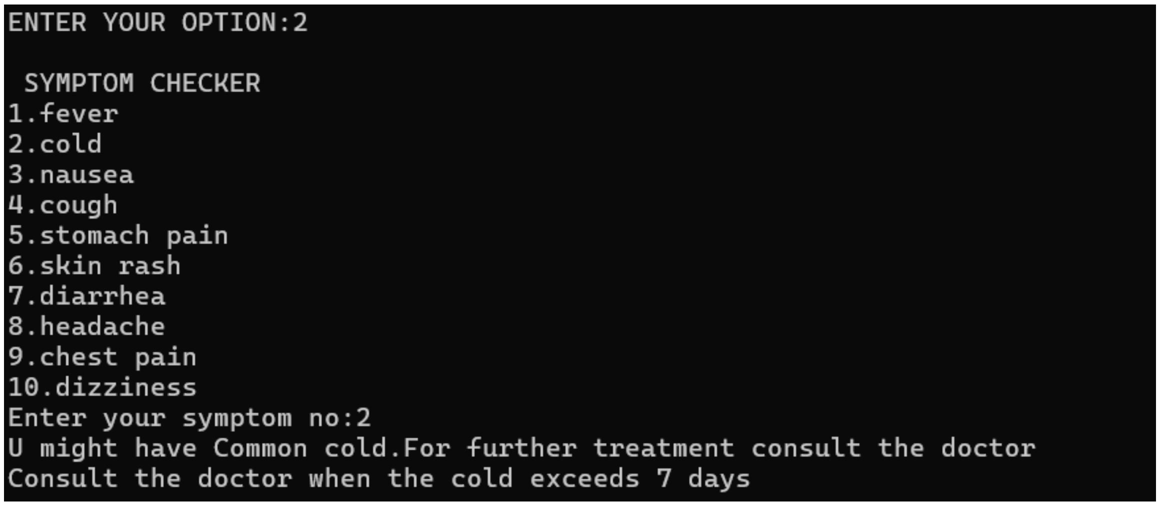
```
 HOME PAGE
1.BMICALCULATOR
2.SYMPTOM CHECKER
3.AGE GROUP ANALYZER
4.APPOINTMENT GENERATOR
5.LOGOUT
ENTER YOUR OPTION:1
```

5.bmicalculator()



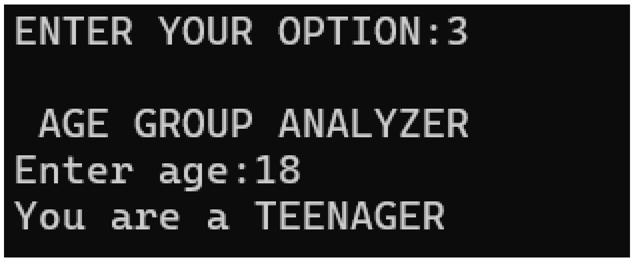
```
ENTER YOUR OPTION:1

 BMI CALCULATOR
Enter height in meters:1.65
Enter weight in kg:55
YOUR BMI IS: 20.202020202020204
YOU ARE NORMAL WEIGHT
```

6.symptomchecker()



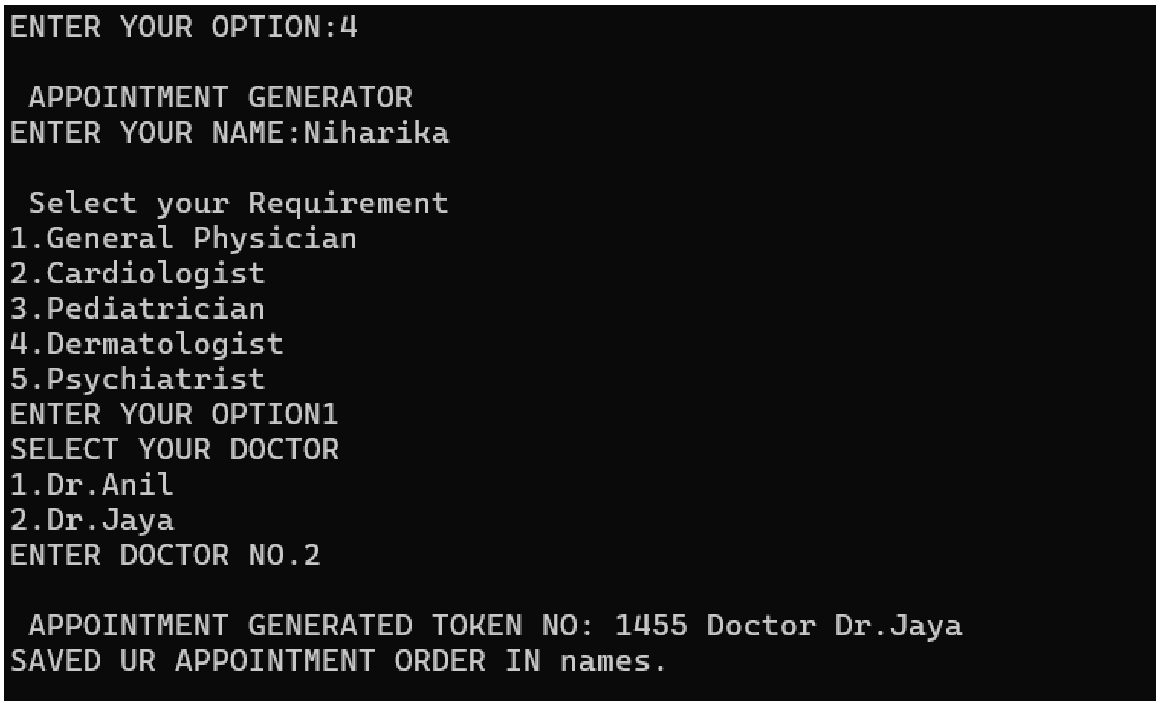
```
ENTER YOUR OPTION:2

 SYMPTOM CHECKER
1.fever
2.cold
3.nausea
4.cough
5.stomach pain
6.skin rash
7.diarrhea
8.headache
9.chest pain
10.dizziness
Enter your symptom no:2
U might have Common cold.For further treatment consult the doctor
Consult the doctor when the cold exceeds 7 days
```

7.ageanalyzer()



```
ENTER YOUR OPTION:3

 AGE GROUP ANALYZER
Enter age:18
You are a TEENAGER
```

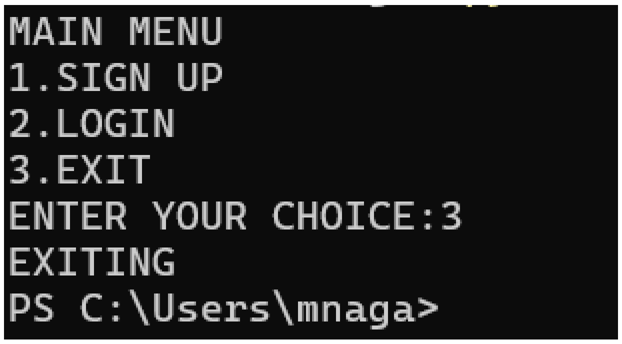
8.appointmentgenerator()



```
ENTER YOUR OPTION:4

 APPOINTMENT GENERATOR
ENTER YOUR NAME:Niharika

 Select your Requirement
1.General Physician
2.Cardiologist
3.Pediatrician
4.Dermatologist
5.Psychiatrist
ENTER YOUR OPTION1
SELECT YOUR DOCTOR
1.Dr.Anil
2.Dr.Jaya
ENTER DOCTOR NO.2

 APPOINTMENT GENERATED TOKEN NO: 1455 Doctor Dr.Jaya
SAVED UR APPOINTMENT ORDER IN names.
```

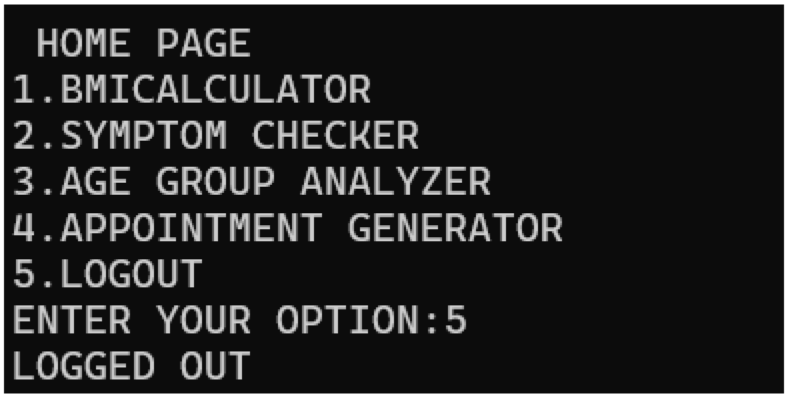
9.exiting

AT MAIN MENY



```
MAIN MENU
1.SIGN UP
2.LOGIN
3.EXIT
ENTER YOUR CHOICE:3
EXITING
PS C:\Users\mnaga>
```

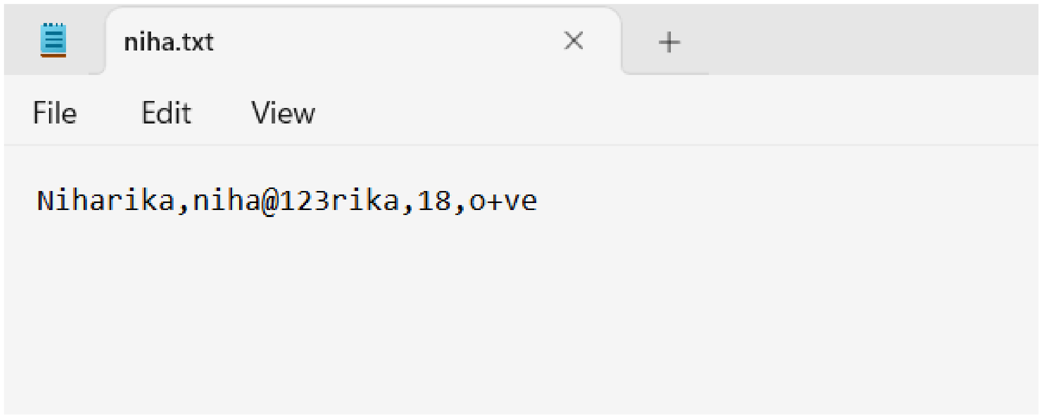
AT HOME PAGE



```
 HOME PAGE
1.BMICALCULATOR
2.SYMPTOM CHECKER
3.AGE GROUP ANALYZER
4.APPOINTMENT GENERATOR
5.LOGOUT
ENTER YOUR OPTION:5
LOGGED OUT
```

10.Text files

For Usernames and Passwords

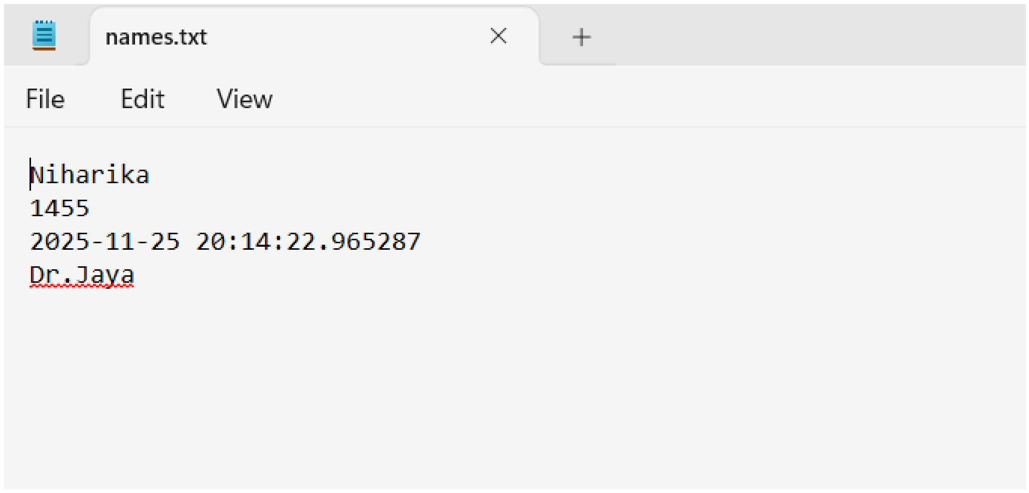


niha.txt

File Edit View

```
Niharika,niha@123rika,18,o+ve
```

For Appointment order



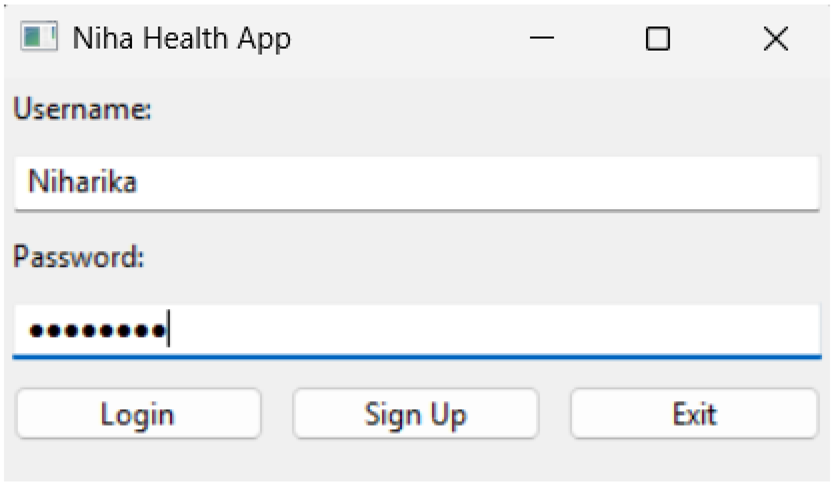
names.txt

File Edit View

```
Niharika
1455
2025-11-25 20:14:22.965287
Dr.Jaya
```

GUI Interface(APP)

1.main()

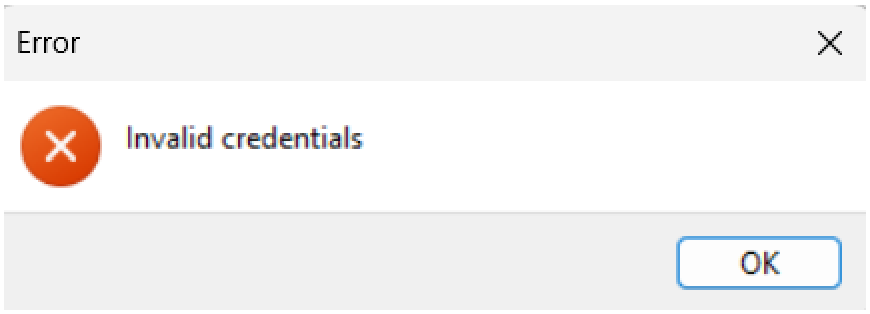

Niha Health App

Username:

Niharika

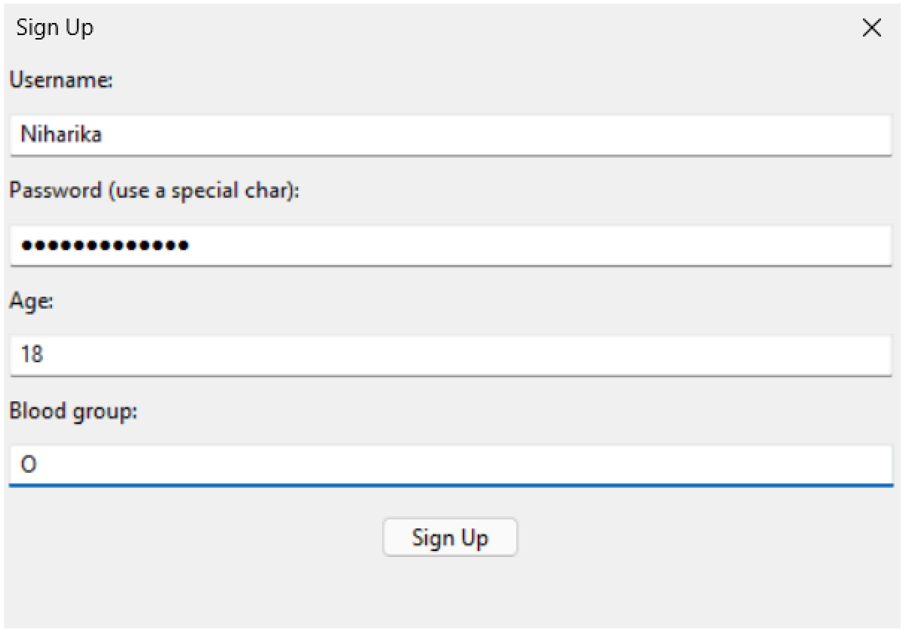
Password:

••••••••


Error

Invalid credentials

2.Signup()


Sign Up

Username:

Niharika

Password (use a special char):

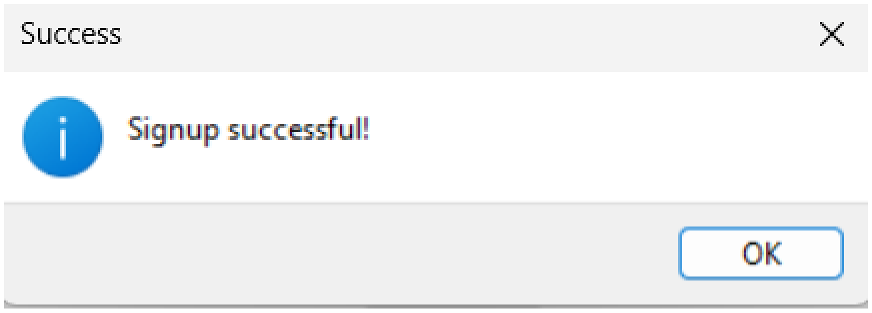
••••••••••••••

Age:

18

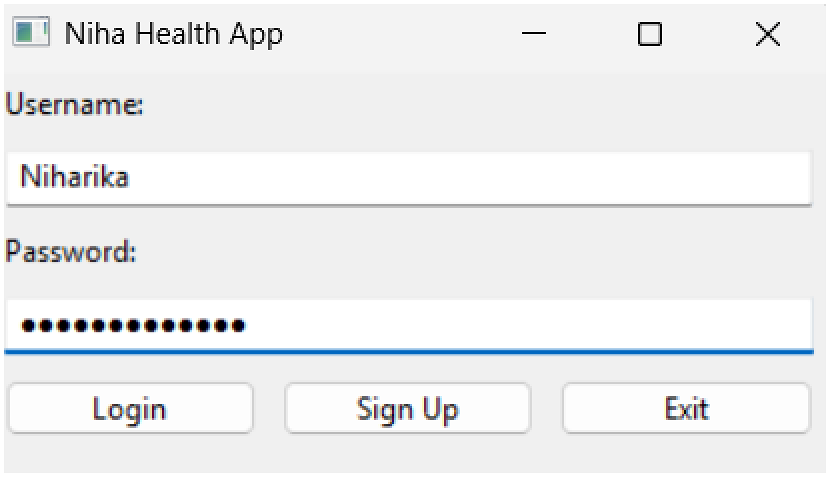
Blood group:

O


Success

Signup successful!

3.Login()

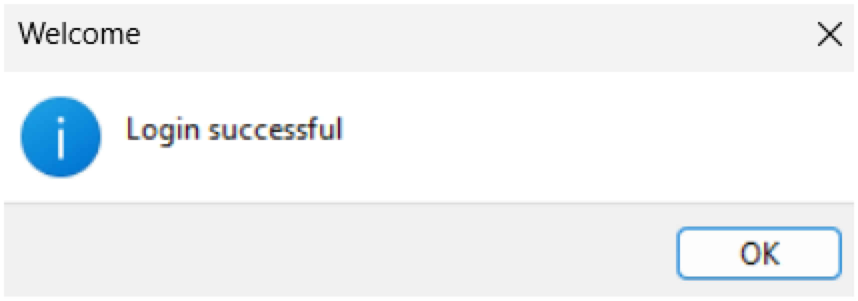

Niha Health App

Username:

Niharika

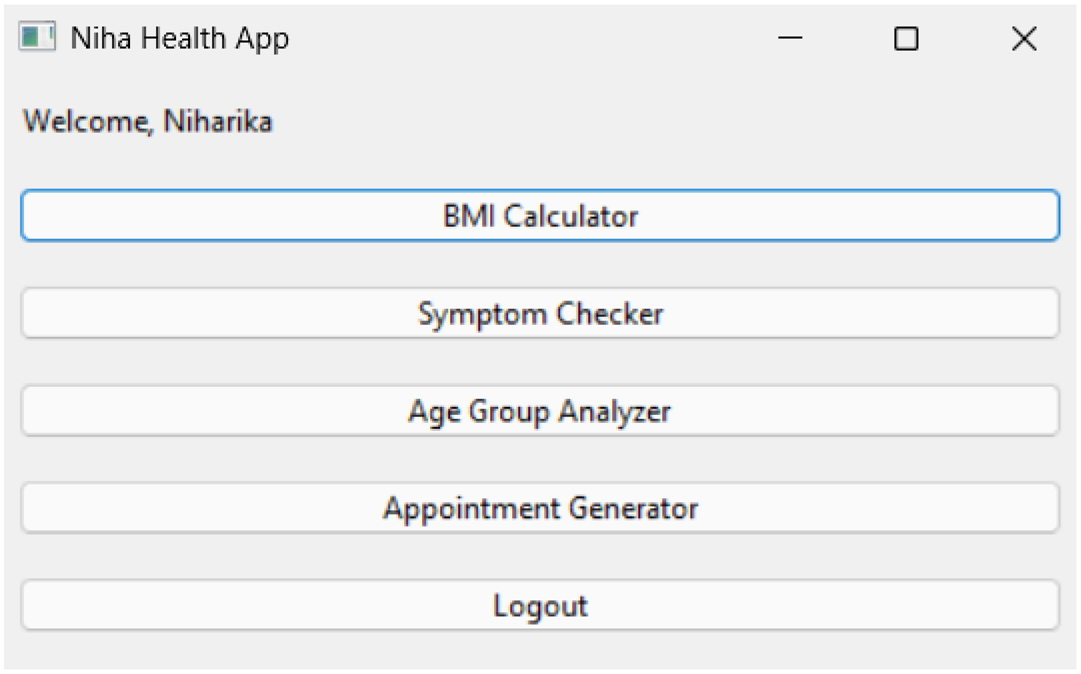
Password:

••••••••••••••


Welcome

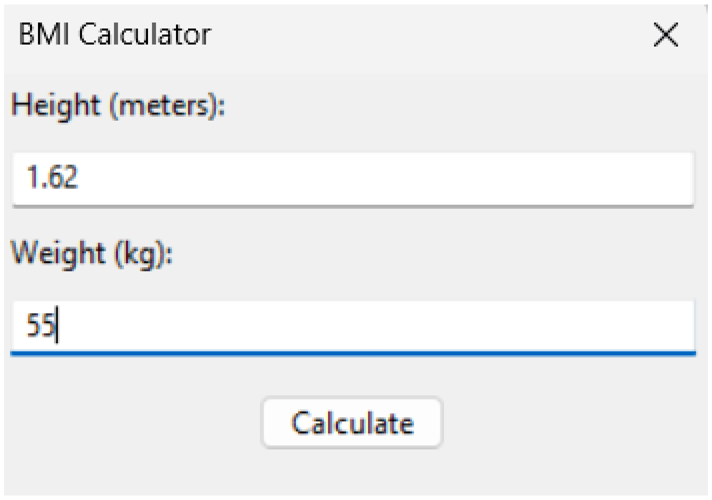
Login successful

4.Homepage

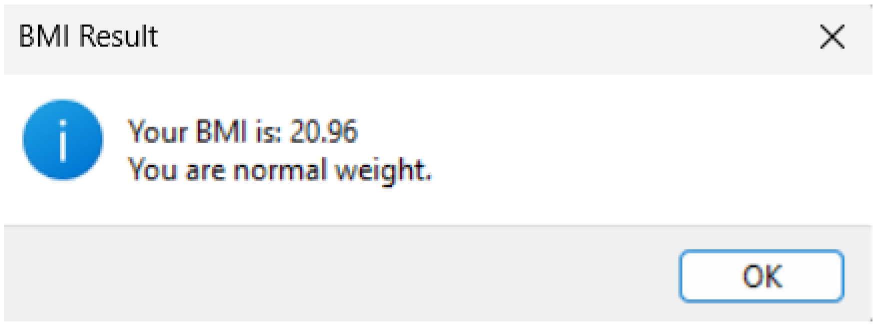


Welcome, Niharika

5.BMI Calculator

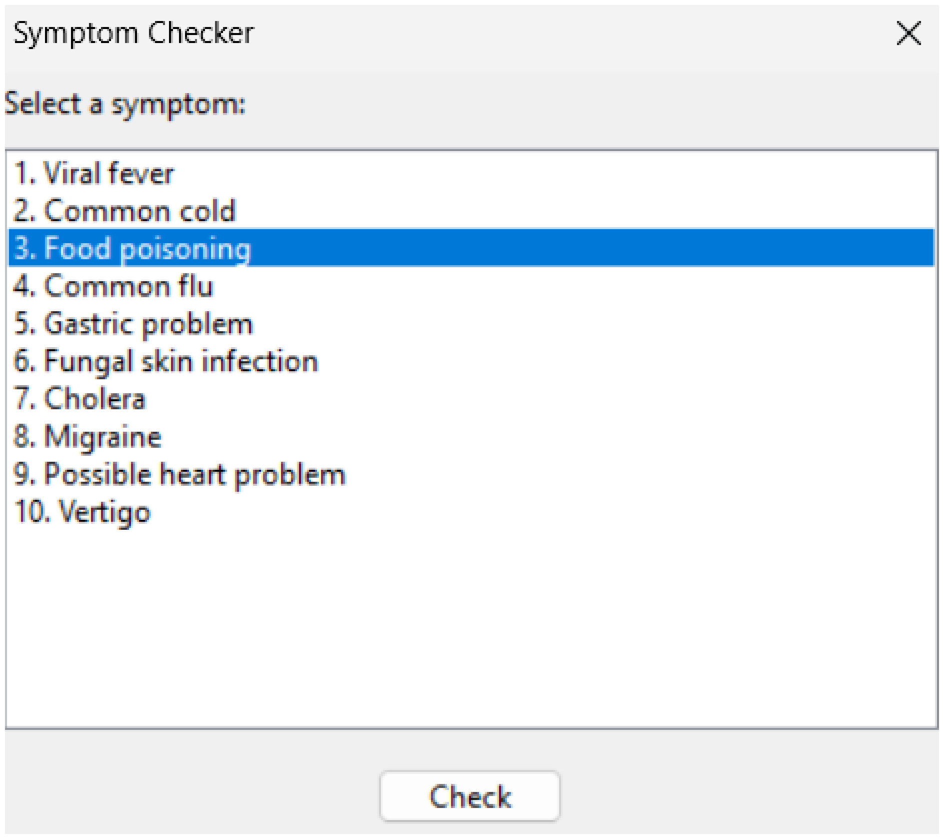


Height (meters):
1.62
Weight (kg):
55



Your BMI is: 20.96
You are normal weight.

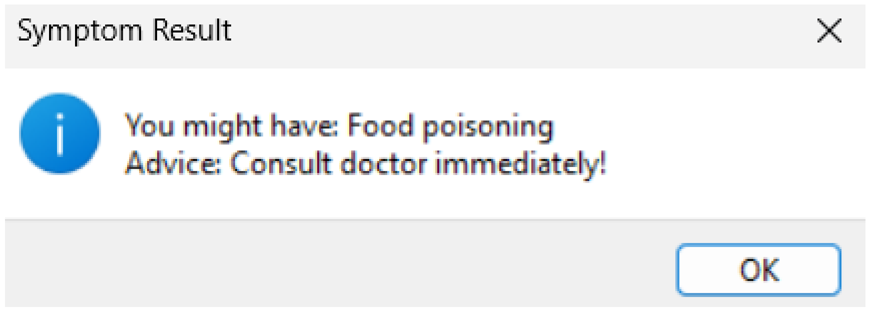
6.Symptom checker



Symptom Checker

Select a symptom:

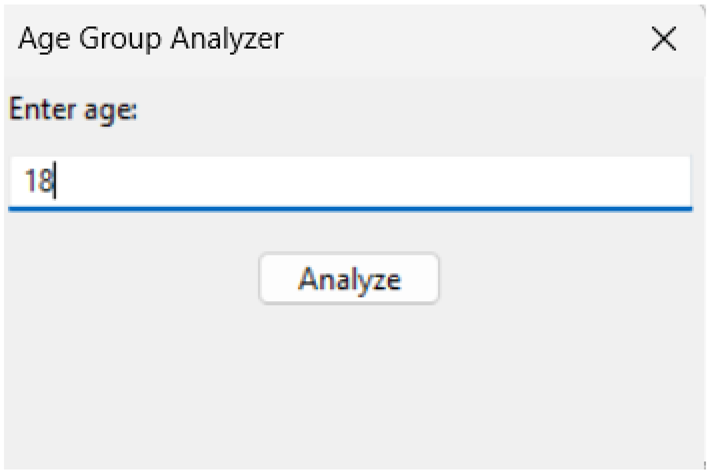
1. Viral fever
2. Common cold
3. Food poisoning
4. Common flu
5. Gastric problem
6. Fungal skin infection
7. Cholera
8. Migraine
9. Possible heart problem
10. Vertigo



Symptom Result

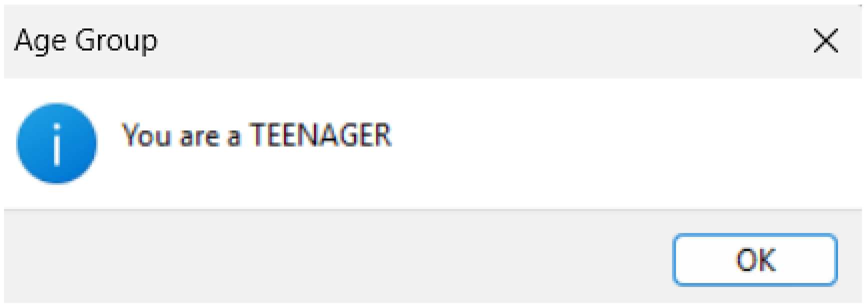
You might have: Food poisoning
Advice: Consult doctor immediately!

7.Age Group Analyzer


Age Group Analyzer

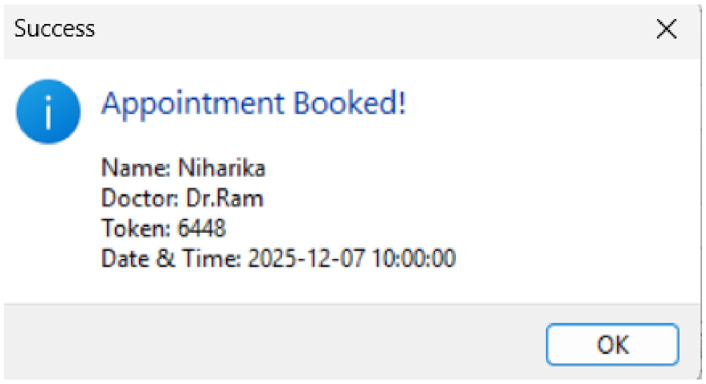
Enter age:

18


Age Group

You are a TEENAGER

8.Appointment Generator


Success

Appointment Booked!

Name: Niharika
Doctor: Dr.Ram
Token: 6448
Date & Time: 2025-12-07 10:00:00

CHALLENGES FACED

Our project was based on healthcare which had implications in storing the doctors and patients details using csv files .But after much trials we had to compromise with text files and simple if-else logic to execute the program.

We had to make sure about the signup and login page since it was accepting login of an unsigned user. Which we could fix successfully .We made sure the signed persons credentials match completely while login to process them into the homepage.

We had to make our project more presentable so we started to add more features by learning new topics which include the ‘re module’ for setting a stronger password which accepts special characters and a match of alphanumeric values.

Apart from all these we had to bring a code that almost works as an application in python script.So a lot of research was needed to present our first project.

This could be said to be a simple healthcare project which includes basic features.We aligned them for a proper use. We couldn't fetch some additional details that could elaborate our project so we stuck to simple program code to run it.

SCOPE FOR IMPROVEMENT

Although the project successfully performs basic healthcare tasks such as BMI calculation ,symptom based suggestions ,age group analysis,and appointment generation there is significant scope for further improvement to make the system more accurate , reliable and user friendly.

The symptom checker can be expanded with additional symptoms ,multiple input operations ,and more precise health suggestions.The BMI calculator can be enhanced by adding features such as ideal weight range,lifestyle tips,or fitness recommendations.

The appointment generator does not provide a realistic mode of application like doctor availability ,time slot selection and automatic confirmation messages. The age analyzer can also be extended to include health risks or suggestions based on age groups.

Adding better input validation ,error handling and smoother navigation would further improve user experience.With these enhancements ,the project can grow into more advanced and practical healthcare applications.

CONCLUSION

In this mini project, we learned how python can be used to build a basic health care system. By taking simple inputs from the user and processing them with conditions and functions, the program is able to give helpful results like BMI, symptom-based suggestions, and basic health advice.

This project helped me understand how programming can solve real-life problems and how important it is in fields like health care. Even though this is a simple model, it shows that python can be used to create more advanced health applications in the future. Overall, this project improved my coding skills and gave me confidence to develop better programs.

BIBLIOGRAPHY

THIS PROJECT HAS TAKEN REFERENCE FROM-

- PESU ACADEMY SLIDES

  https://www.pesuacademy.com/Academy/s/studentProfilePESU

  UNIT1 ,UNIT2 AND UNIT3

- GEEKS FOR GEEKS

  GeeksforGeeks | Your All-in-One Learning Portal

- How to Read a Text file In Python Effectively
- datetime — Basic date and time types — Python 3.14.0 documentation

THANKYOU