Group No -- 16

PROJECT REPORT: PYTHON HEALTH TRACKER

Team Members:
1)Dhyeyaa Palrecha - PES1UG25EC084
2)Dvaipaayan.P.M - PES1UG25EE013
3)Tejas.K - PES1UG25EC283
4)Dushyanth - PES1UG25AM122

1. Problem Statement :

The objective of this project is to develop a user-friendly, desktop-based application that assists users in monitoring their physical health. Standard command-line tools are often difficult for non-technical users to operate. Our solution, HealthMetrics Pro, provides a Graphical User Interface (GUI) to calculate Body Mass Index (BMI) and estimate Body Fat Percentage using the U.S. Navy Method. It addresses the need for long-term tracking by automatically saving all user records to a persistent CSV database.

2. Approach & Methodology:

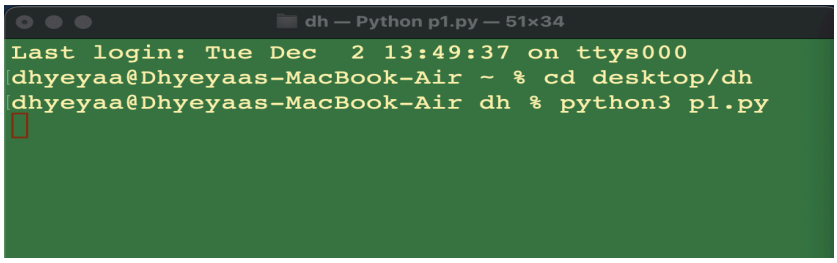
We developed the application using Python with the following key components:

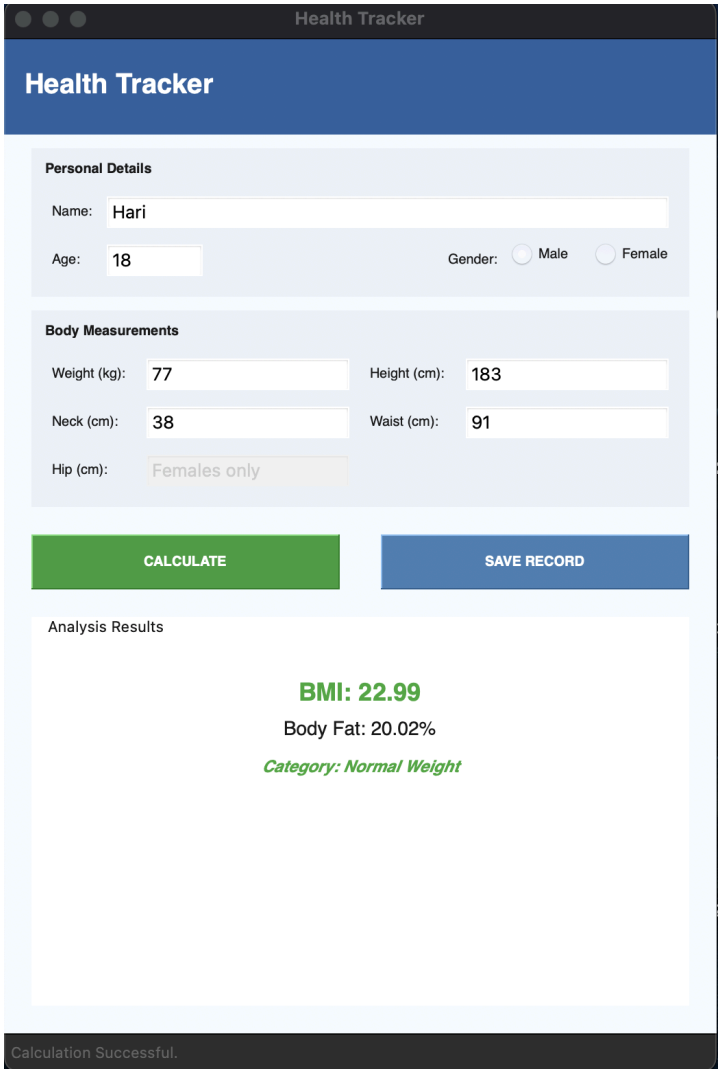
- Graphical User Interface (GUI): We utilized the `wxPython` library to create a robust windowed application. The layout was managed using `wx.BoxSizer` and `wx.GridBagSizer` to ensure the interface looks clean and organized.
- Mathematical Logic: The application implements the U.S. Navy Body Fat formula, which requires logarithmic calculations (`math.log10`) involving waist, neck, and height measurements. The logic automatically distinguishes between Male and Female formulas.
- Cross-Platform Compatibility: We implemented custom logic to detect and enforce specific text and background colors. This ensures the application remains visible and usable regardless of whether the user's computer is in Light Mode or Dark Mode.
- Data Persistence: The app uses the `csv` module to append new records to a `health_history.csv` file, automatically creating the file with headers if it does not exist.

3. Sample Input/Output

- Input:
  - Name: Hari
  - Age: 18
  - Weight: 77 kg, Height: 173 cm
  - Neck: 38 cm, Waist: 91 cm
- Output (Displayed on Screen):
  - BMI: 22.99
  - Body Fat: 20.02%
  - Category: Normal Weight (Displayed in Green)

- Persistence: A confirmation popup appears: "Record saved successfully to health_history.csv".





health_history

Date	Name	Age	Gender	Weight (kg)	Height (m)	Waist (cm)	Neck (cm)	Hip (cm)	BMI	Body Fat %	Category
2025-12-01 09:10:23	Hari	18	M	77.0	1.83	91.0	38.0	0.0	22.99	20.02	Normal Weight
2025-12-01 09:14:59	Krish	19	M	72.4	1.785	96.52	38.0	0.0	22.72	24.47	Normal Weight
2025-12-01 09:16:13	Tejas	18	M	53.0	1.64	71.0	28.0	0.0	19.71	15.62	Normal Weight
2025-12-01 09:18:27	navneet	17	M	83.0	1.71	101.6	44.0	0.0	28.38	25.19	Overweight
2025-12-01 09:22:02	Ramesh	18	M	60.0	1.8	66.0	30.0	0.0	18.52	6.61	Normal Weight
2025-12-01 09:27:40	Pooja	18	F	50.0	1.62	60.0	26.0	69.0	19.05	8.54	Normal Weight
2025-12-01 09:31:06	Diya	18	F	75.0	1.65	86.0	38.0	90.0	27.55	27.43	Overweight
2025-12-01 09:34:52	A	18	M	80.0	1.79	98.0	30.0	0.0	24.97	30.11	Obese
2025-12-01 09:40:17	Hari	18	M	77.0	1.83	91.0	38.0	0.0	22.99	20.02	Normal Weight
2025-12-02 10:22:57	tej	18	F	53.0	1.64	85.0	38.0	90.0	19.71	27.19	Normal Weight

4. Challenges Faced

- Theme Consistency: A major challenge was that input text would turn invisible (white-on-white) when the computer was in Dark Mode. We resolved this by creating a helper function that forces specific foreground (Black) and background (White) colors for all input fields.
- GUI Rendering: We encountered a "race condition" on macOS where the window would appear blank upon launching. We fixed this by using `wx.CallAfter(self.Layout)`, which forces the application to re-draw the layout immediately after the window opens.

5. Scope for Improvement

- Data Visualization: We could integrate `matplotlib` to generate trend lines showing how the user's weight has changed over the past months.
- User Profiles: Implementing a login system to support multiple distinct users with separate history files.

6.Code

```
import wx
import wx.lib.buttons as buttons
import csv
import os
import math
from datetime import datetime

FILE_NAME = "health_history.csv"

def calculate_bmi(weight_kg, height_m):
    try:
        bmi = weight_kg / (height_m ** 2)
        return round(bmi, 2)
    except ZeroDivisionError:
        return 0.0

def calculate_body_fat_navy(gender, height_cm, waist_cm, neck_cm, hip_cm=0):
    try:
        if gender.lower() == 'm':
            if waist_cm - neck_cm <= 0: return 0.0
            log_wn = math.log10(waist_cm - neck_cm)
            log_h = math.log10(height_cm)
            bfp = 495 / (1.0324 - 0.19077 * log_wn + 0.15456 * log_h) - 450
        else:
            if waist_cm + hip_cm - neck_cm <= 0: return 0.0
            log_whn = math.log10(waist_cm + hip_cm - neck_cm)
            log_h = math.log10(height_cm)
            bfp = 495 / (1.29579 - 0.35004 * log_whn + 0.22100 * log_h) - 450
        return round(max(bfp, 0), 2)
    except Exception:
        return 0.0

def get_bmi_category(bmi):
    if bmi < 18.5: return "Underweight"
    elif 18.5 <= bmi < 24.9: return "Normal Weight"
    elif 25 <= bmi < 29.9: return "Overweight"
    else: return "Obese"
```

```
def save_to_csv(data):
    file_exists = os.path.isfile(FILE_NAME)
    try:
        with open(FILE_NAME, mode='a', newline='') as file:
            writer = csv.writer(file)
            if not file_exists:
                writer.writerow(["Date", "Name", "Age", "Gender", "Weight (kg)", "Height (m)", "Waist (cm)", "Neck (cm)", "Hip (cm)", "BMI", "Body
Fat %", "Category"])
            writer.writerow(data)
        return True, f"Record saved successfully to {FILE_NAME}"
    except Exception as e:
        return False, str(e)

class HealthTrackerFrame(wx.Frame):
    def __init__(self):
        style = wx.DEFAULT_FRAME_STYLE & ~(wx.RESIZE_BORDER | wx.MAXIMIZE_BOX)
        super().__init__(parent=None, title='Health Tracker', size=(520, 780), style=style)

        self.bg_color = wx.Colour(245, 250, 255)
        self.panel_bg = wx.Colour(235, 242, 250)
        self.text_color = wx.Colour(20, 20, 20)
        self.header_color = wx.Colour(45, 100, 160)

        self.font_std = wx.Font(10, wx.FONTFAMILY_SWISS, wx.FONTSTYLE_NORMAL, wx.FONTWEIGHT_NORMAL)
        self.font_bold = wx.Font(10, wx.FONTFAMILY_SWISS, wx.FONTSTYLE_NORMAL, wx.FONTWEIGHT_BOLD)

        self.InitUI()
        self.Centre()
        self.Show()

        wx.CallAfter(self.Layout)
        wx.CallAfter(self.Refresh)

    def create_label(self, parent, text):
        lbl = wx.StaticText(parent, label=text)
        lbl.SetForegroundColour(self.text_color)
        lbl.SetFont(self.font_std)
        return lbl

    def create_text_input(self, parent, size=wx.DefaultSize, value=""):
        txt = wx.TextCtrl(parent, size=size, value=value)
        txt.SetForegroundColour(wx.BLACK)
        txt.SetBackgroundColour(wx.WHITE)
        return txt
```

```
    def InitUI(self):
        panel = wx.Panel(self)
        panel.SetBackgroundColour(self.bg_color)

        main_sizer = wx.BoxSizer(wx.VERTICAL)

        header_panel = wx.Panel(panel, size=(-1, 70))
        header_panel.SetBackgroundColour(self.header_color)
        header_sizer = wx.BoxSizer(wx.HORIZONTAL)

        header_title = wx.StaticText(header_panel, label="Health Tracker")
        header_title.SetFont(wx.Font(20, wx.FONTFAMILY_SWISS, wx.FONTSTYLE_NORMAL, wx.FONTWEIGHT_BOLD))
        header_title.SetForegroundColour(wx.WHITE)

        header_sizer.Add(header_title, 0, wx.CENTER | wx.ALL, 15)
        header_panel.SetSizer(header_sizer)

        main_sizer.Add(header_panel, 0, wx.EXPAND)

        content_sizer = wx.BoxSizer(wx.VERTICAL)

        group1_panel = wx.Panel(panel)
        group1_panel.SetBackgroundColour(self.panel_bg)
        group1_border = wx.BoxSizer(wx.VERTICAL)

        lbl_p = wx.StaticText(group1_panel, label="Personal Details")
        lbl_p.SetFont(self.font_bold)
        lbl_p.SetForegroundColour(self.text_color)
        group1_border.Add(lbl_p, 0, wx.LEFT | wx.TOP, 10)

        grid1 = wx.GridBagSizer(vgap=12, hgap=10)

        grid1.Add(self.create_label(group1_panel, "Name:"), pos=(0, 0), flag=wx.ALIGN_CENTER_VERTICAL)
        self.input_name = self.create_text_input(group1_panel)
        grid1.Add(self.input_name, pos=(0, 1), span=(1, 3), flag=wx.EXPAND)

        grid1.Add(self.create_label(group1_panel, "Age:"), pos=(1, 0), flag=wx.ALIGN_CENTER_VERTICAL)
        self.input_age = self.create_text_input(group1_panel, size=(70, -1))
        grid1.Add(self.input_age, pos=(1, 1))

        grid1.Add(self.create_label(group1_panel, "Gender:"), pos=(1, 2), flag=wx.ALIGN_CENTER_VERTICAL|wx.ALIGN_RIGHT)
        gender_box = wx.BoxSizer(wx.HORIZONTAL)
```

```
        self.radio_male = wx.RadioButton(group1_panel, label="", style=wx.RB_GROUP)
        lbl_male = wx.StaticText(group1_panel, label="Male")
        lbl_male.SetForegroundColour(self.text_color)
        lbl_male.SetFont(self.font_std)

        self.radio_female = wx.RadioButton(group1_panel, label="")
        lbl_female = wx.StaticText(group1_panel, label="Female")
        lbl_female.SetForegroundColour(self.text_color)
        lbl_female.SetFont(self.font_std)

        lbl_male.Bind(wx.EVT_LEFT_DOWN, lambda evt: self.manual_radio_toggle('m'))
        lbl_female.Bind(wx.EVT_LEFT_DOWN, lambda evt: self.manual_radio_toggle('f'))

        gender_box.Add(self.radio_male, 0, wx.ALIGN_CENTER_VERTICAL)
        gender_box.Add(lbl_male, 0, wx.ALIGN_CENTER_VERTICAL | wx.LEFT | wx.RIGHT, 5)
        gender_box.Add(self.radio_female, 0, wx.ALIGN_CENTER_VERTICAL | wx.LEFT, 15)
        gender_box.Add(lbl_female, 0, wx.ALIGN_CENTER_VERTICAL | wx.LEFT, 5)

        grid1.Add(gender_box, pos=(1, 3))

        grid1.AddGrowableCol(1)

        group1_border.Add(grid1, 1, wx.EXPAND | wx.ALL, 15)
        group1_panel.SetSizer(group1_border)

        content_sizer.Add(group1_panel, 0, wx.EXPAND | wx.ALL, 10)

        group2_panel = wx.Panel(panel)
        group2_panel.SetBackgroundColour(self.panel_bg)
        group2_border = wx.BoxSizer(wx.VERTICAL)

        lbl_b = wx.StaticText(group2_panel, label="Body Measurements")
        lbl_b.SetFont(self.font_bold)
        lbl_b.SetForegroundColour(self.text_color)
        group2_border.Add(lbl_b, 0, wx.LEFT | wx.TOP, 10)

        grid2 = wx.GridBagSizer(vgap=12, hgap=15)

        grid2.Add(self.create_label(group2_panel, "Weight (kg):"), pos=(0, 0), flag=wx.ALIGN_CENTER_VERTICAL)
        self.input_weight = self.create_text_input(group2_panel)
        grid2.Add(self.input_weight, pos=(0, 1), flag=wx.EXPAND)

        grid2.Add(self.create_label(group2_panel, "Height (cm):"), pos=(0, 2), flag=wx.ALIGN_CENTER_VERTICAL)
        self.input_height = self.create_text_input(group2_panel)
        grid2.Add(self.input_height, pos=(0, 3), flag=wx.EXPAND)
```

```
        grid2.Add(self.create_label(group2_panel, "Neck (cm):"), pos=(1, 0), flag=wx.ALIGN_CENTER_VERTICAL)
        self.input_neck = self.create_text_input(group2_panel)
        grid2.Add(self.input_neck, pos=(1, 1), flag=wx.EXPAND)

        grid2.Add(self.create_label(group2_panel, "Waist (cm):"), pos=(1, 2), flag=wx.ALIGN_CENTER_VERTICAL)
        self.input_waist = self.create_text_input(group2_panel)
        grid2.Add(self.input_waist, pos=(1, 3), flag=wx.EXPAND)

        grid2.Add(self.create_label(group2_panel, "Hip (cm):"), pos=(2, 0), flag=wx.ALIGN_CENTER_VERTICAL)
        self.input_hip = self.create_text_input(group2_panel, value="Females only")
        self.input_hip.SetForegroundColour(wx.Colour(150, 150, 150))
        self.input_hip.Enable(False)
        grid2.Add(self.input_hip, pos=(2, 1), flag=wx.EXPAND)

        grid2.AddGrowableCol(1)
        grid2.AddGrowableCol(3)

        group2_border.Add(grid2, 1, wx.EXPAND | wx.ALL, 15)
        group2_panel.SetSizer(group2_border)

        content_sizer.Add(group2_panel, 0, wx.EXPAND | wx.LEFT | wx.RIGHT, 10)

        main_sizer.Add(content_sizer, 0, wx.EXPAND | wx.LEFT | wx.RIGHT, 10)

        btn_sizer = wx.BoxSizer(wx.HORIZONTAL)

        self.btn_calc = buttons.GenButton(panel, label="CALCULATE")
        self.btn_calc.SetBackgroundColour(wx.Colour(50, 160, 60))
        self.btn_calc.SetForegroundColour(wx.WHITE)
        self.btn_calc.SetFont(self.font_bold)
        self.btn_calc.SetBezelWidth(1)
        self.btn_calc.SetMinSize((150, 40))

        self.btn_save = buttons.GenButton(panel, label="SAVE RECORD")
        self.btn_save.SetBackgroundColour(wx.Colour(240, 240, 240))
        self.btn_save.SetForegroundColour(wx.Colour(150, 150, 150))
        self.btn_save.SetFont(self.font_bold)
        self.btn_save.SetBezelWidth(1)
        self.btn_save.SetMinSize((150, 40))
        self.btn_save.Enable(False)

        btn_sizer.Add(self.btn_calc, 1, wx.RIGHT, 15)
        btn_sizer.Add(self.btn_save, 1, wx.LEFT, 15)

        main_sizer.Add(btn_sizer, 0, wx.EXPAND | wx.ALL, 20)
```

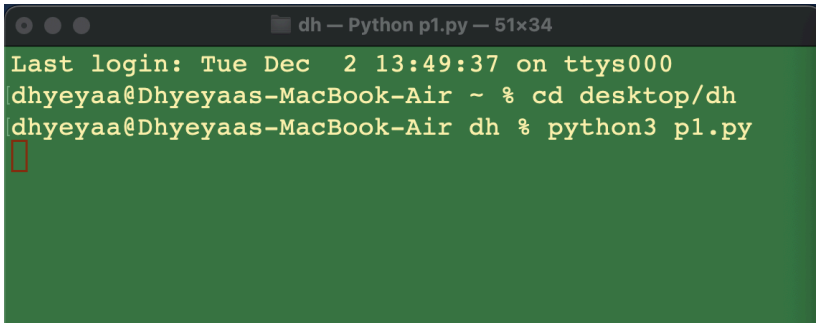
```
        self.result_container = wx.Panel(panel)
        self.result_container.SetBackgroundColour(wx.Colour(255, 255, 255))

        res_border_sizer = wx.StaticBoxSizer(wx.VERTICAL, self.result_container, "Analysis Results")
        res_border_sizer.GetStaticBox().SetForegroundColour(self.text_color)
        res_border_sizer.GetStaticBox().SetFont(self.font_bold)

        res_sizer = wx.BoxSizer(wx.VERTICAL)

        self.lbl_bmi_val = wx.StaticText(self.result_container, label="BMI: --")
        self.lbl_bmi_val.SetFont(wx.Font(18, wx.FONTFAMILY_SWISS, wx.FONTSTYLE_NORMAL, wx.FONTWEIGHT_BOLD))
        self.lbl_bmi_val.SetForegroundColour(self.text_color)

        self.lbl_bfp_val = wx.StaticText(self.result_container, label="Body Fat: --%")
        self.lbl_bfp_val.SetFont(wx.Font(14, wx.FONTFAMILY_SWISS, wx.FONTSTYLE_NORMAL, wx.FONTWEIGHT_NORMAL))
        self.lbl_bfp_val.SetForegroundColour(self.text_color)

        self.lbl_cat = wx.StaticText(self.result_container, label="Category: --")
        self.lbl_cat.SetFont(wx.Font(12, wx.FONTFAMILY_SWISS, wx.FONTSTYLE_ITALIC, wx.FONTWEIGHT_BOLD))
        self.lbl_cat.SetForegroundColour(self.text_color)

        res_sizer.Add(self.lbl_bmi_val, 0, wx.CENTER | wx.TOP, 25)
        res_sizer.Add(self.lbl_bfp_val, 0, wx.CENTER | wx.TOP, 10)
        res_sizer.Add(self.lbl_cat, 0, wx.CENTER | wx.TOP | wx.BOTTOM, 15)

        res_border_sizer.Add(res_sizer, 1, wx.EXPAND)
        self.result_container.SetSizer(res_border_sizer)

        main_sizer.Add(self.result_container, 1, wx.EXPAND | wx.LEFT | wx.RIGHT | wx.BOTTOM, 20)

        self.statusbar = self.CreateStatusBar()
        self.statusbar.SetStatusText("Ready")

        panel.SetSizer(main_sizer)

        self.Bind(wx.EVT_RADIOBUTTON, self.on_gender_toggle, self.radio_male)
        self.Bind(wx.EVT_RADIOBUTTON, self.on_gender_toggle, self.radio_female)
        self.Bind(wx.EVT_BUTTON, self.on_calculate, self.btn_calc)
        self.Bind(wx.EVT_BUTTON, self.on_save, self.btn_save)

        frame_sizer = wx.BoxSizer(wx.VERTICAL)
        frame_sizer.Add(panel, 1, wx.EXPAND)
        self.SetSizer(frame_sizer)
```

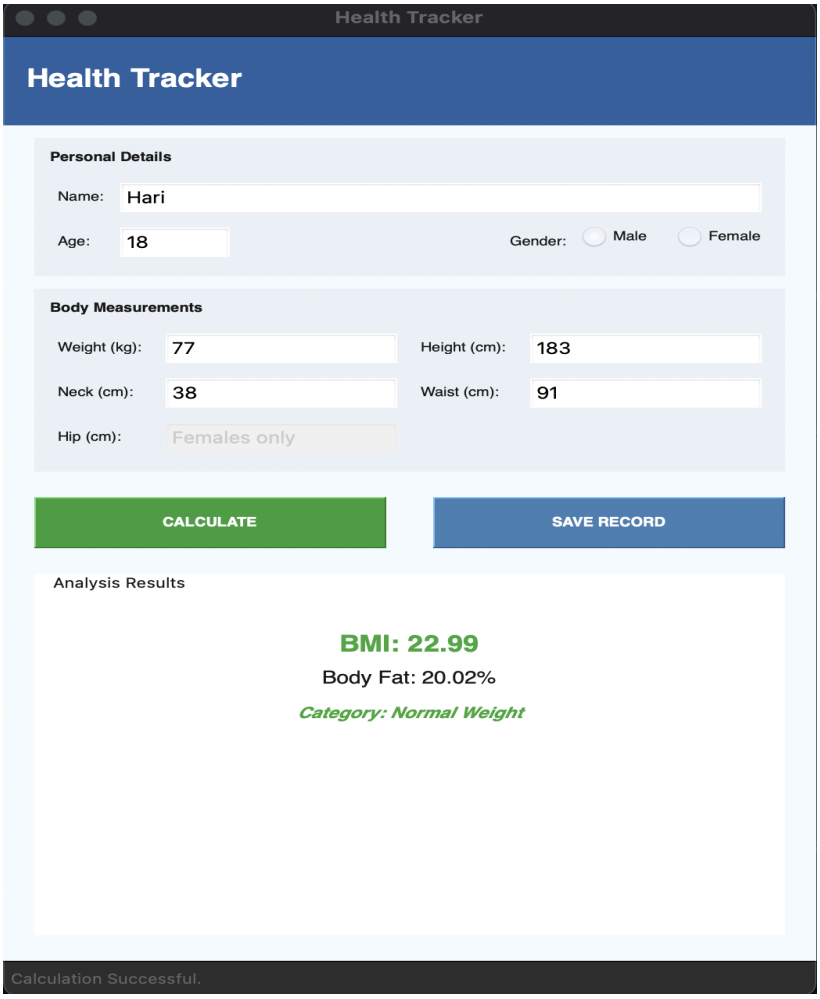
```
    def manual_radio_toggle(self, gender):
        if gender == 'm':
            self.radio_male.SetValue(True)
            self.on_gender_toggle(None)
        else:
            self.radio_female.SetValue(True)
            self.on_gender_toggle(None)

    def on_gender_toggle(self, event):
        if self.radio_female.GetValue():
            self.input_hip.Enable(True)
            self.input_hip.SetBackgroundColour(wx.WHITE)
            if self.input_hip.GetValue() == "Females only":
                self.input_hip.SetValue("")
                self.input_hip.SetForegroundColour(wx.BLACK)
        else:
            self.input_hip.Enable(False)
            self.input_hip.SetValue("Females only")
            self.input_hip.SetForegroundColour(wx.Colour(150, 150, 150))
            self.input_hip.SetBackgroundColour(wx.Colour(240, 240, 240))
        self.input_hip.Refresh()

    def on_calculate(self, event):
        try:
            name = self.input_name.GetValue()
            if not name:
                wx.MessageBox("Please enter a name.", "Missing Info", wx.ICON_WARNING)
                return

            age_val = self.input_age.GetValue()
            weight_val = self.input_weight.GetValue()
            height_val = self.input_height.GetValue()
            neck_val = self.input_neck.GetValue()
            waist_val = self.input_waist.GetValue()

            if not (age_val and weight_val and height_val and neck_val and waist_val):
                wx.MessageBox("Please fill in all fields.", "Missing Info", wx.ICON_WARNING)
                return

            age = int(age_val)
            weight = float(weight_val)
            height_cm = float(height_val)
            neck = float(neck_val)
            waist = float(waist_val)

            gender = 'f' if self.radio_female.GetValue() else 'm'
```

```
            hip = 0.0
            if gender == 'f':
                hip_val = self.input_hip.GetValue()
                if not hip_val or hip_val == "Females only":
                    wx.MessageBox("Hip measurement is required for females.", "Missing Info", wx.ICON_WARNING)
                    return
                hip = float(hip_val)

            height_m = height_cm / 100
            bmi = calculate_bmi(weight, height_m)
            bfp = calculate_body_fat_navy(gender, height_cm, waist, neck, hip)
            category = get_bmi_category(bmi)

            self.lbl_bmi_val.SetLabel(f"BMI: {bmi}")
            self.lbl_bfp_val.SetLabel(f"Body Fat: {bfp}%")
            self.lbl_cat.SetLabel(f"Category: {category}")

            if category == "Normal Weight":
                color = wx.Colour(40, 160, 40)
            elif category == "Overweight":
                color = wx.Colour(220, 120, 0)
            else:
                color = wx.Colour(220, 40, 40)

            self.lbl_bmi_val.SetForegroundColour(color)
            self.lbl_cat.SetForegroundColour(color)
            self.result_container.Layout()

            timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
            self.current_data = [timestamp, name, age, gender.upper(), weight, height_m, waist, neck, hip, bmi, bfp, category]

            self.btn_save.Enable(True)
            self.btn_save.SetBackgroundColour(wx.Colour(70, 130, 180))
            self.btn_save.SetForegroundColour(wx.WHITE)
            self.btn_save.Refresh()

            self.statusbar.SetStatusText("Calculation Successful.")

        except ValueError:
            wx.MessageBox("Please enter valid numeric values.", "Input Error", wx.ICON_ERROR)
```

```
    def on_save(self, event):
        if hasattr(self, 'current_data'):
            success, msg = save_to_csv(self.current_data)
            self.statusbar.SetStatusText(msg)
            if success:
                wx.MessageBox(msg, "Saved", wx.ICON_INFORMATION)
                self.btn_save.Enable(False)
                self.btn_save.SetBackgroundColour(wx.Colour(240, 240, 240))
                self.btn_save.SetForegroundColour(wx.Colour(150, 150, 150))
                self.btn_save.Refresh()

if __name__ == '__main__':
    app = wx.App()
    frame = HealthTrackerFrame()
    app.MainLoop()
```

7.Input And Output





health_history

	Date	Name	Age	Gender	Weight (kg)	Height (m)	Waist (cm)	Neck (cm)	Hip (cm)	BMI	Body Fat %	Category
2	2025-12-01 09:10:23	Hari	18	M	77.0	1.83	91.0	38.0	0.0	22.99	20.02	Normal Weight
3	2025-12-01 09:14:59	Krish	19	M	72.4	1.785	96.52	38.0	0.0	22.72	24.47	Normal Weight
4	2025-12-01 09:16:13	Tejas	18	M	53.0	1.64	71.0	28.0	0.0	19.71	15.62	Normal Weight
5	2025-12-01 09:18:27	navneet	17	M	83.0	1.71	101.6	44.0	0.0	28.38	25.19	Overweight
6	2025-12-01 09:22:02	Ramesh	18	M	60.0	1.8	66.0	30.0	0.0	18.52	6.61	Normal Weight
7	2025-12-01 09:27:40	Pooja	18	F	50.0	1.62	60.0	26.0	69.0	19.05	8.54	Normal Weight
8	2025-12-01 09:31:06	Diya	18	F	75.0	1.65	86.0	38.0	90.0	27.55	27.43	Overweight
9	2025-12-01 09:34:52	A	18	M	80.0	1.79	98.0	30.0	0.0	24.97	30.11	Obese
10	2025-12-01 09:40:17	Hari	18	M	77.0	1.83	91.0	38.0	0.0	22.99	20.02	Normal Weight
11	2025-12-02 10:22:57	tej	18	F	53.0	1.64	85.0	38.0	90.0	19.71	27.19	Normal Weight

Exiting program.Stay healthy!

THANK YOU!!!!