Project Report

Body Mass Index (BMI) Calculator using Python.

Submitted to: Oasis Infobyte

Submitted by: Shreejay Balasaheb Jadhav

1. Introduction :

This project aims to develop a comprehensive Body Mass Index (BMI) calculator application using Python. The application will empower users to easily calculate their BMI based on their weight and height. Leveraging the tkinter library, we will create an intuitive graphical user interface (GUI) that facilitates seamless user interaction.

Beyond basic BMI calculation, the application will incorporate a robust data storage mechanism using SQLite. This allows users to maintain a history of their BMI measurements over time. Furthermore, to provide valuable insights into BMI trends, we will integrate Matplotlib for generating clear and informative visualizations.

2. Objectives :

The primary objectives of this project are:

1. To develop a BMI calculator with an intuitive GUI.
2. To store user data in a SQLite database for tracking and retrieval.
3. To visualize BMI trends using Matplotlib.
4. To ensure robust error handling and user-friendly features for seamless operation.

3. Literature Review

Project Goals :

To create a user-friendly application that calculates Body Mass Index (BMI), stores user data in a database, and provides graphical visualizations of BMI trends.

Project Components:

1. **Python:** The core programming language for the application's logic and functionality.
2. **tkinter:** A standard Python library for creating graphical user interfaces (GUIs).

3. **Matplotlib:** A plotting library for generating visualizations of BMI data.
4. **SQLite:** A lightweight, file-based database for storing user data persistently.

The application will feature the following functionalities:

- **BMI Calculation:** Accurate BMI calculation based on user-provided weight and height.
- **User-Friendly GUI:** An intuitive graphical interface built with tkinter for easy data input and result display.
- **Data Storage:** Persistent data storage using SQLite to maintain a history of BMI measurements.
- **Data Visualization:** Generation of line graphs and other relevant plots using Matplotlib to visualize BMI trends over time.
- **Data Retrieval:** The ability to retrieve and display historical BMI records.
- **Error Handling:** Robust error handling to ensure data validity and application stability.

By combining these technologies, this project aims to provide a practical and insightful tool for users to monitor and understand their BMI trends effectively.

Detailed breakdown of the components:

1. **Python:**

- Serves as the foundation for the entire application.
- Handles the mathematical calculations for BMI.
- Manages data flow between the GUI, database, and plotting library.

2. **tkinter:**

- Creates the interactive user interface with input fields (weight, height), buttons (calculate, save, view history), and output displays (BMI result, category).
- Provides a visually appealing and easy-to-navigate application.

3. **Matplotlib:**

- Generates line graphs to visualize BMI trends over time, showing how BMI changes with each measurement.
- Can create histograms to show the distribution of BMI.
- Allows users to gain a visual understanding of their BMI history.

4. **SQLite:**

- Stores user data, including weight, height, BMI, date, and time, in a local database file.
- Ensures data persistence, allowing users to access their historical BMI records whenever needed.
- Simple database to implement.

Detailed Explanation :

Database Setup Functions :

create_database()

- Creates a SQLite database (bmi_data.db).
- If the table (bmi_records) doesn't exist, it creates one with columns:
    - (id) : Primary key, auto-increment.
    - (username) : Name of the user.
    - (weight) : User's weight.
    - (height) : User's height.
    - (bmi) : Calculated BMI.
    - (category) : BMI health category.
    - (date) : Timestamp of entry.

insert_data(username, weight, height, bmi, category)

- Inserts a record into the bmi_records table with the given parameters.

fetch_user_data(username)

- Retrieves all BMI records for a given username.

BMI Calculation Functions :

- Computes BMI using the formula:

$$BMI = \frac{\text{weight (kg)}}{\text{height (m)^2}}$$

- Categorizes BMI into:
  - **Underweight** (< 18.5)
  - **Normal weight** (18.5 - 24.9)
  - **Overweight** (25 - 29.9)
  - **Obesity** (≥ 30)

- Returns the BMI and its category.

validate_inputs(weight, height)

- Validates that the weight and height are positive numbers.
- Displays an error message using messagebox.showerror for invalid inputs.

calculate_and_store()

- Retrieves user input for username, weight, and height.
- Validates inputs using validate_inputs.
- Calculates BMI and category using calculate_bmi.
- Displays the result.
- Saves the result to the database using insert_data.
- Refreshes the history table.

History and Trend Functions :

refresh_history()

- Fetches all records for the username from the database.
- Clears the history table (history_tree) and populates it with records.

plot_bmi_trend()

- Fetches records for username.
- Extracts BMI values and corresponding dates.

- Creates a line plot using Matplotlib to show BMI trends.
- Displays the plot in a new Toplevel window.

GUI Components :

- Main Window Setup :

- root = tk.Tk(): Initializes the main application window.
- root.title("BMI Calculator"): Sets the title of the application.
- root.geometry("600x500"): Defines the window size.

- User Input Fields :

- username_label and username_entry: Fields for the username.
- weight_label and weight_entry: Fields for entering weight.
- height_label and height_entry: Fields for entering height.
  Buttons
- calculate_button: Button to calculate BMI and store the data.
- trend_button: Button to view the BMI trend.

- Result Display :

- result_label: Displays the calculated BMI and its category.

- History Section :

- history_frame: Container for the history table.
- history_tree: Treeview widget for displaying previous BMI records.

- Database Initialization :

- create_database(): Ensures the database and table are set up before running the app.

Matplotlib Integration :

- Uses FigureCanvasTkAgg to embed Matplotlib graphs into Tkinter windows.

Conclusion :

This project successfully integrates Python's capabilities with Tkinter, SQLite, and Matplotlib to create a comprehensive BMI calculator. The application provides users with an intuitive interface to calculate BMI, store historical records, and visualize trends, thereby enabling better health monitoring. By combining these technologies, the application achieves a balance between functionality and usability, making it a practical tool for personal health management.