VIT BHOPAL

vityarthi
YOUR LEARNING DESTINATION

# Evaluated Course Project

## Health Tracking System


**Submitted by-**

**Lakshya jain**

**Registration no.**

**25BAI10994**

# Index

# Introduction

The Health Tracker is a lightweight Python desktop application developed using Tkinter, Python's standard GUI library. It enables users to enter and view essential daily health information such as date, steps taken, calories burned, and workout duration. The system stores all records in a simple text file, making it easy to use, portable, and suitable for beginners. Additionally, the application provides motivational messages to encourage users to maintain consistency in their health routines. This project demonstrates key concepts such as GUI development, file handling, modular programming, and simple data organization, making it an ideal introductory project for understanding real-world software design using Python.

**THE GITHUB REPO IS**

**https://github.com/chupkarlakshya/HealthTrackingSystem**

# Problem Statement

In today's busy routine, people often forget to keep track of their daily health activities such as steps taken, calories burned, and workout duration.

Although many fitness applications exist, they are often:

- Too complicated for beginners
- Require login or internet connection
- Consume large storage or battery
- Filled with unnecessary features

There is a need for a simple, lightweight, and easy-to-use desktop application that allows users to quickly record and review their daily health data without complexity.

The Health Tracker project aims to solve this by providing a minimalistic GUI tool that stores data locally, displays past records, and motivates users to maintain a healthy routine.

# Objectives

The main objectives of the Health Tracker application are:

## Primary Objectives

- **To develop a simple and user-friendly desktop application using Python and Tkinter.**
- To allow users to easily record daily health information such as date, steps, calories, and workout duration.
- To display all stored health records clearly inside the application.
- To provide a feature for viewing the most recent 30 entries (monthly records).

## Additional Objectives

- To motivate users with random positive messages to encourage consistency.
- To demonstrate fundamental programming skills such as:
  - GUI design
  - File handling
  - Data formatting
  - Modular function-based programming

## Educational Objectives

- To apply classroom concepts in a real project.
- To learn how GUI applications interact with storage files.
- To practice debugging, testing, and improving application design.

# Functional Requirements

### 4.1 Add Daily Record

- The user can input:
  - Date (YYYY-MM-DD)
  - Steps taken
  - Calories burned
  - Workout duration (in minutes)
- On clicking Add, the data is saved into the data.txt file in structured format.

### 4.2 View All Records

- Displays every record stored in data.txt.
- Shows information in a clear, labelled format:
  - Date
  - Steps
  - Calories
  - Workout minutes

### 4.3 View Monthly Records

- Displays the most recent 30 entries (representing one month of data).
- Reads the last 150 lines (30 records × 5 lines per record).
- Shows them in the same labelled format.

### 4.4 Motivational Messages

- When the user clicks Motivation, a positive motivational message is displayed.
- Messages are selected randomly from a predefined list.

# Non-Functional Requirements

The Health Tracker application also satisfies several non-functional requirements that ensure quality, usability, and reliability.

**1. Usability**

- Simple and clean Tkinter interface.
- Large input fields and buttons for easy interaction.
- Clear labels for every value (Date, Steps, Calories, Workout).

**2. Performance**

- Very fast execution since storage is file-based.
- Quick load time because the program uses lightweight modules (tkinter, random).

**3. Reliability**

- Handles missing file cases gracefully by showing “No data” instead of crashing.
- Structured data format ensures correct reading and display of records.

**4. Maintainability**

- Code is modular and divided into separate functions:
  - Add()
  - View_Records()
  - Monthly_Records()
  - Motivation()
- Easy to upgrade or extend (e.g., adding graphs or database support).

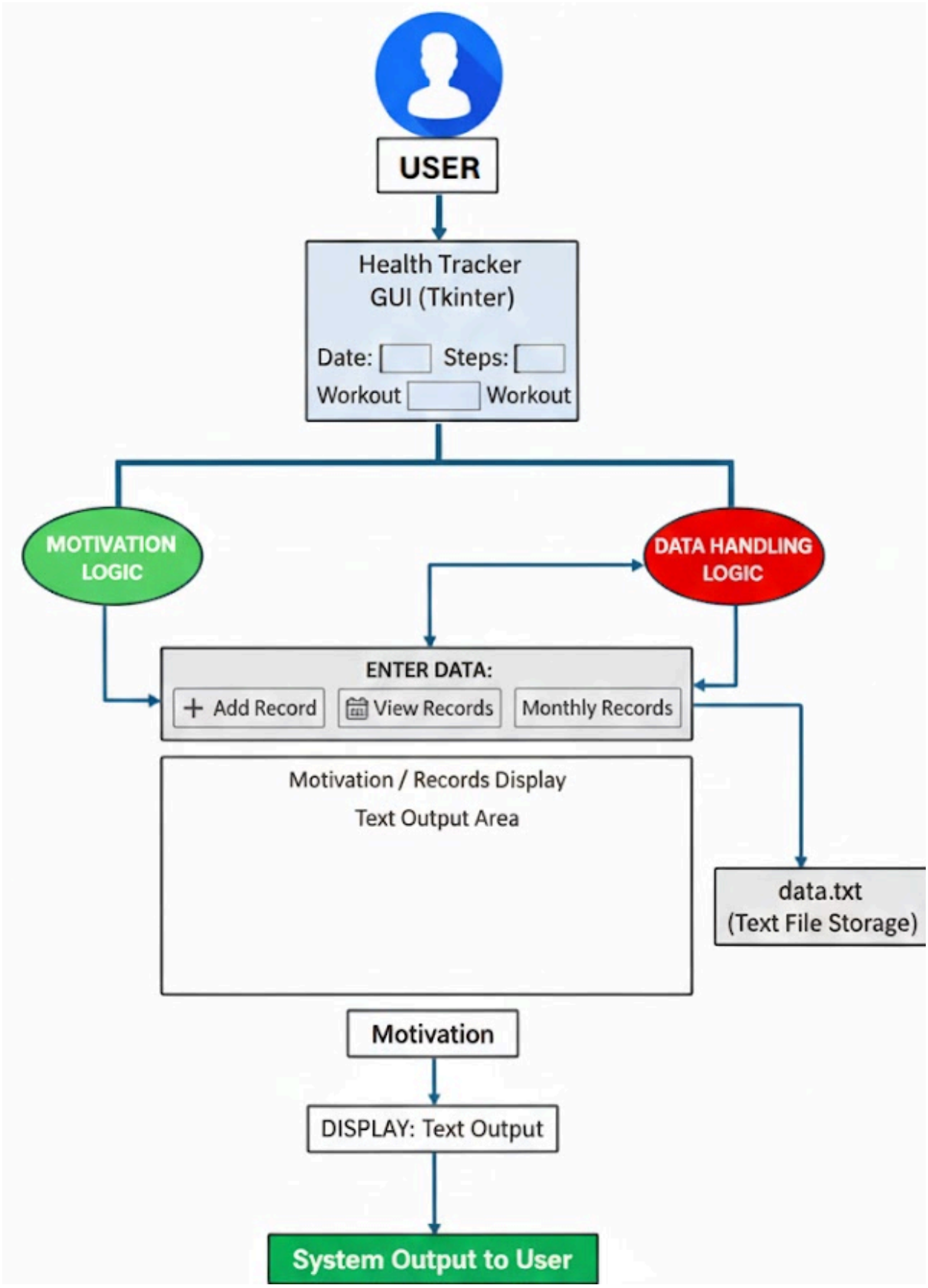
**5. Portability**

- Can run on any system with Python installed (Windows, Mac, Linux).
- Does not require installation of heavy external dependencies.

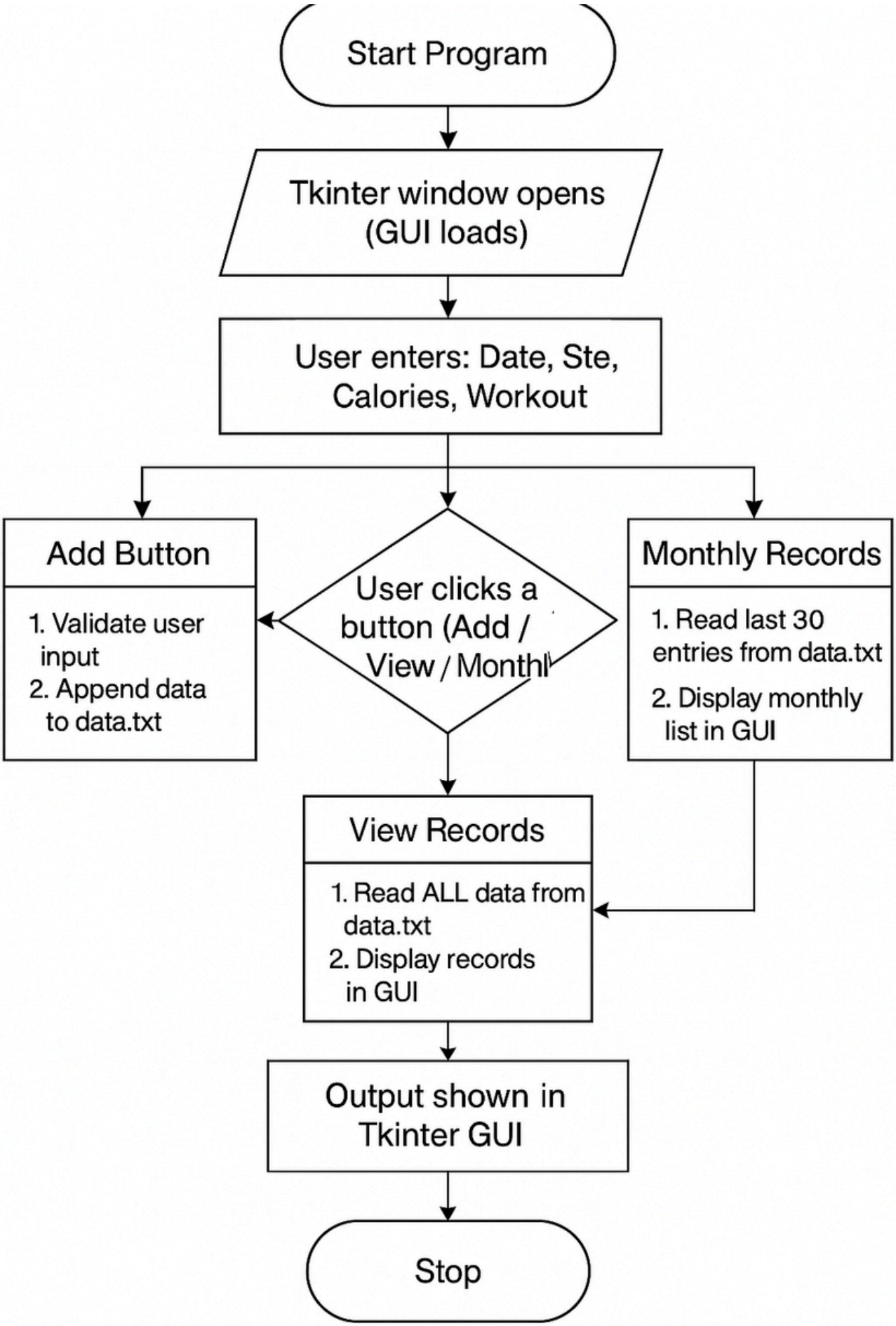
**6. Resource Efficiency**

- Uses only basic Python and Tkinter.
- Storage is a simple text file, requiring almost no memory.

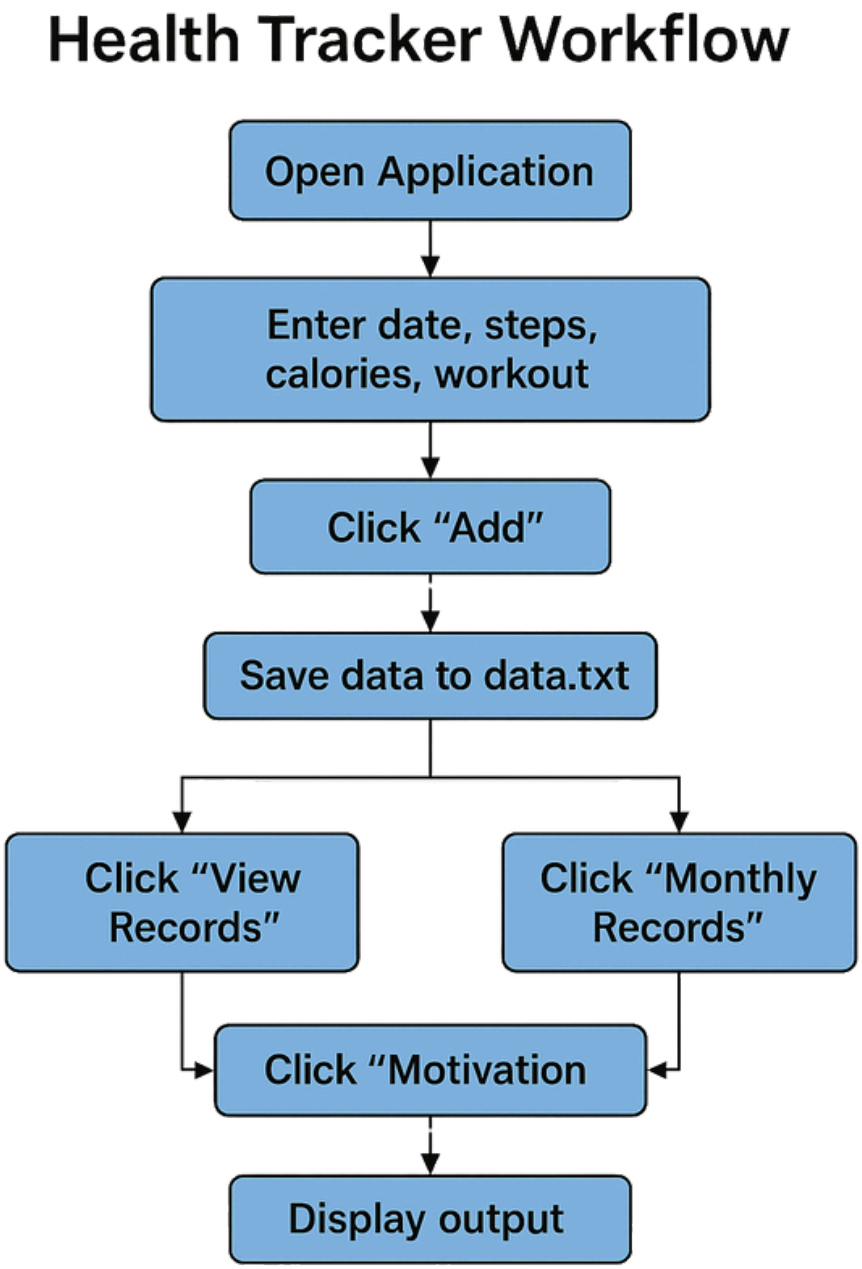
# System Architecture Design



# Workflow Diagram



# Design Diagram



# Decisions & Rationale

**1. Tkinter for GUI**

Decision: Use Tkinter.

Reason: Easy, built-in, perfect for simple desktop apps.

**2. Text File for Storage**

Decision: Save data in data.txt.

Reason: Simple, no database needed, easy to read/write.

**3. Fixed 5-Line Record Format**

Decision: Store each entry as 5 lines.

Reason: Makes reading records easy and avoids complex parsing.

**4. . Motivational Messages**

Decision: Add a Motivation button.

Reason: Makes the app more engaging and encouraging.

**5. Minimal Dependencies**

Decision: Only use tkinter and random.

Reason: Lightweight, portable, no installation issues.

6. Simple UI Layout

Decision: Vertical inputs and buttons.

Reason: Clear, simple, user-friendly.

# Implementation

## A. Core Technologies & Structure

- Language: Python 3.x for simplicity and readability.
- GUI: Tkinter—the standard, built-in library—ensuring cross-platform portability with zero external dependencies.
- Storage: data.txt (Local Text File). Acts as a simple, fast, persistent logbook.
- Structure: A single procedural script organized around four key functions, managing data flow to and from the global data.txt.

## B. GUI Setup & Input Handling

- Inputs: Four tk.Entry widgets capture Date, Steps, Calories, and Workout data.
- Output: A single tk.Text widget (txt) serves as the dynamic display for status messages, record views, and motivational quotes.

## C. Record Addition Logic (Add function)

1. Retrieval: Captures user input using .get().
2. Validation: Simple check (if d and s and c and w:) ensures all four fields are non-empty.
3. File Writing: Opens data.txt in append mode ("a") to preserve existing data.
4. Fixed Format: Each record is strictly written as five separate lines. The fifth line, "END", is the vital record delimiter for easy retrieval.
5. Feedback: Displays "Saved" status in the output console.

## D. Record Viewing Logic (View_Records & Monthly_Records)

- Parsing Strategy: Both functions read all lines and iterate, processing 5 lines at a time (i += 5) to reconstruct each record based on the fixed format.
- Monthly Optimization: The Monthly_Records function efficiently calculates the start point for the last 30 entries (150 lines) using:

$$start = \max(0, \text{len(lines)} - 30 \times 5)$$

- This prevents unnecessary reading or display of older data.

```python
import tkinter as tk              # importing tkinter for GUI
import random                     # for motivational messages

fname = "data.txt"                # file where data will be saved


# --------------------- MAIN WINDOW ---------------------

win = tk.Tk()                     # creating main window
win.title("Health Tracker")       # window title


# --------------------- INPUT FIELDS ---------------------

tk.Label(win, text="Date (YYYY-MM-DD)").pack()
date_entry = tk.Entry(win)
date_entry.pack()

tk.Label(win, text="Steps").pack()
steps_entry = tk.Entry(win)
steps_entry.pack()

tk.Label(win, text="Calories").pack()
cal_entry = tk.Entry(win)
cal_entry.pack()

tk.Label(win, text="Workout (minutes)").pack()
work_entry = tk.Entry(win)
work_entry.pack()

txt = tk.Text(win)                # output text box
txt.pack()
```

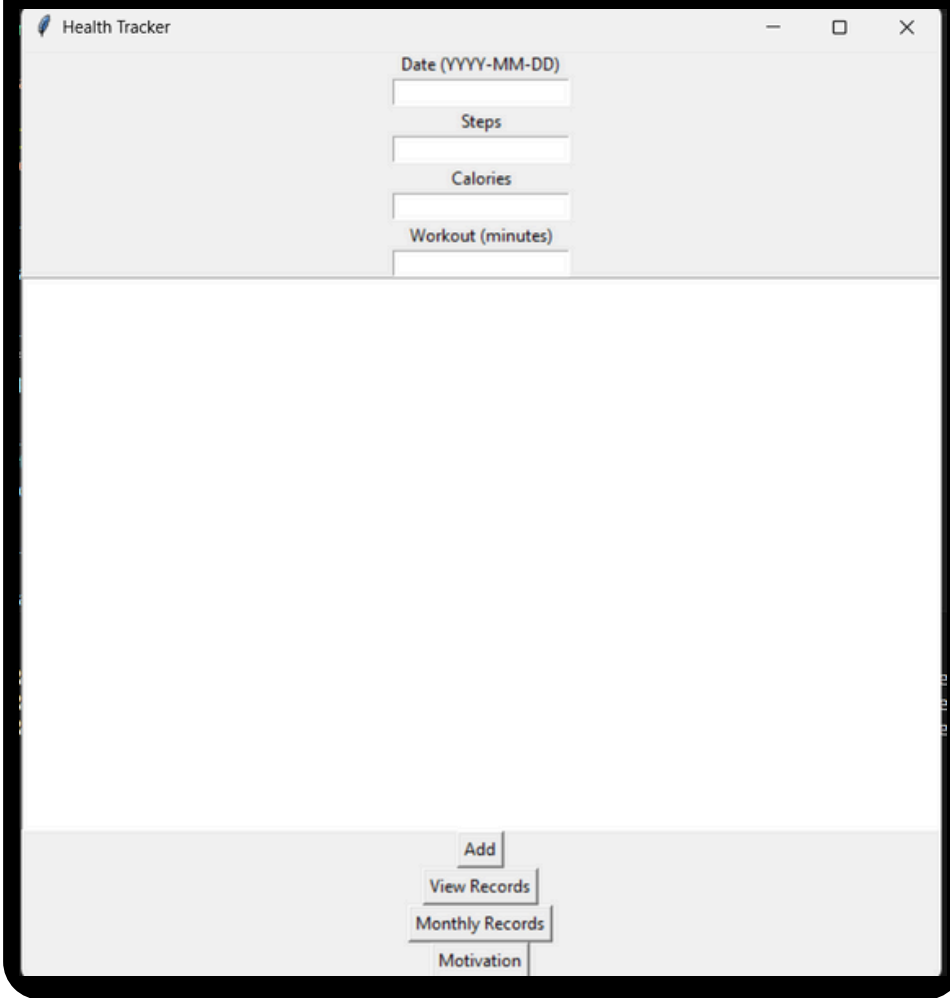
```
# -------------------- ADD RECORD --------------------
def Add():
    d = date_entry.get()          # reading date
    s = steps_entry.get()         # reading steps
    c = cal_entry.get()           # reading calories
    w = work_entry.get()          # reading workout minutes

    if d and s and c and w:       # checking all fields filled
        f = open(fname, "a")      # opening file in append mode
        f.write(d + "\n")         # writing date
        f.write(s + "\n")         # writing steps
        f.write(c + "\n")         # writing calories
        f.write(w + "\n")         # writing workout minutes
        f.write("END\n")          # marking end of record
        f.close()

        txt.delete("1.0", tk.END)
        txt.insert(tk.END, "Saved\n")   # showing message

tk.Button(win, text="Add", command=Add).pack()   # add button

# -------------------- VIEW ALL RECORDS --------------------
def View_Records():
    txt.delete("1.0", tk.END)           # clearing display

    try:
        lines = open(fname).readlines() # reading file lines
    except:
        txt.insert(tk.END, "No data\n") # file not found
        return

    record = []                         # temporary list for one record

    for line in lines:
        line = line.strip()

        if line == "END":               # one full record completed
            if len(record) == 4:
                txt.insert(tk.END, "Date: " + record[0] + "\n")
                txt.insert(tk.END, "Steps: " + record[1] + "\n")
                txt.insert(tk.END, "Calories: " + record[2] + "\n")
                txt.insert(tk.END, "Workout: " + record[3] + " minutes\n")
                txt.insert(tk.END, "------------------\n")
            record = []                 # reset for next record
        else:
            record.append(line)         # adding data to record list
```

```
tk.Button(win, text="View Records", command=View_Records).pack()  # view button


# ---------------------- VIEW LAST 30 RECORDS ----------------------

def Monthly_Records():
    txt.delete("1.0", tk.END)           # clearing display

    try:
        lines = open(fname).readlines() # reading file
    except:
        txt.insert(tk.END, "No data\n")
        return

    records = []                        # all stored records
    record = []

    for line in lines:
        line = line.strip()

        if line == "END":
            if len(record) == 4:
                records.append(record)  # storing one full record
            record = []
        else:
            record.append(line)

    last30 = records[-30:]              # selecting last 30 entries

    if not last30:
        txt.insert(tk.END, "Not enough data\n")
        return

    for r in last30:                    # displaying last 30 entries
        txt.insert(tk.END, "Date: " + r[0] + "\n")
        txt.insert(tk.END, "Steps: " + r[1] + "\n")
        txt.insert(tk.END, "Calories: " + r[2] + "\n")
        txt.insert(tk.END, "Workout: " + r[3] + " minutes\n")
        txt.insert(tk.END, "--------------------\n")


tk.Button(win, text="Monthly Records", command=Monthly_Records).pack()  # button
```

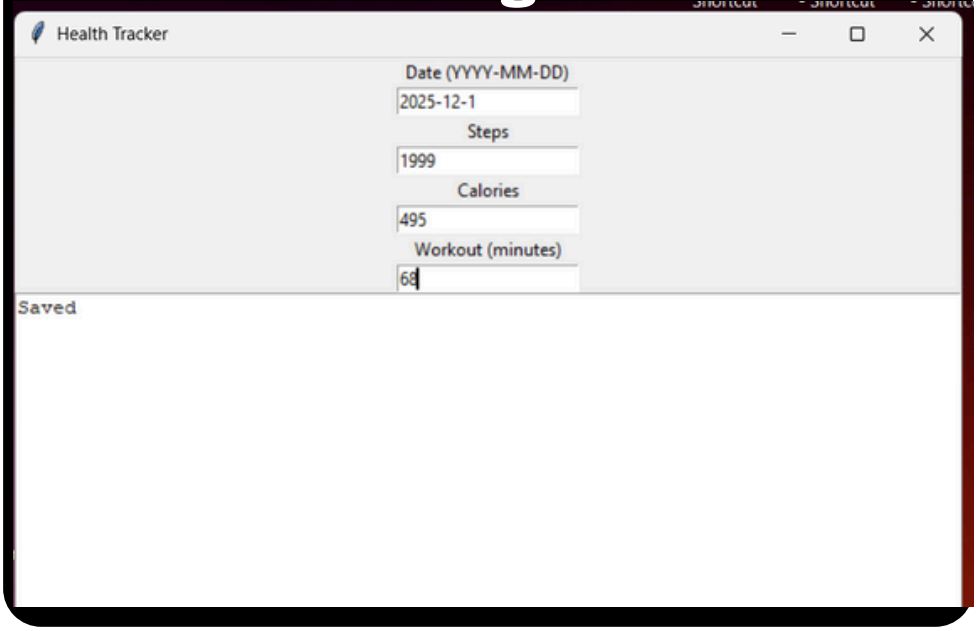
```
# ---------------------- MOTIVATION FEATURE ----------------------

messages = [
    "Great job! Keep going!",
    "Small steps every day make big changes!",
    "You are improving! Keep it up!",
    "You can do this!",
    "Proud of you for tracking your health!",
    "Stay active and stay healthy!",
    "Consistency wins!",
    "Believe in yourself!"
]                                   # motivational lines


def Motivation():
    txt.delete("1.0", tk.END)       # clear output box
    msg = random.choice(messages)   # randomly picking one message
    txt.insert(tk.END, msg + "\n")  # displaying it


tk.Button(win, text="Motivation", command=Motivation).pack()   # button


# ---------------------- START APPLICATION ----------------------

win.mainloop()                      # running the app window
```

# Screenshots

## Main Window UI



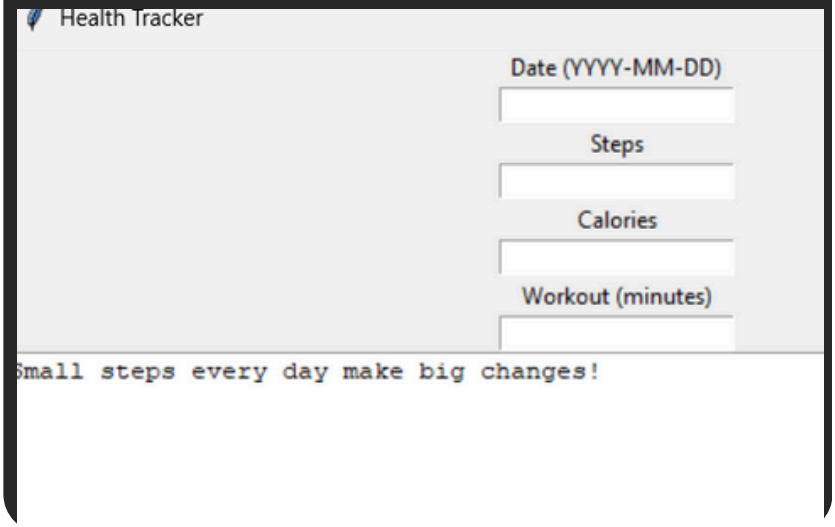

## Entering Data



## Past Records

```
Calories: ds
Workout: sdsdwefewfew minutes
--------------------
Date: 2025-12-1
Steps: 1999
Calories: 495
Workout: 68 minutes
--------------------
Date: 2025-12-2
Steps: 1954
Calories: 394
Workout: 40 minutes
--------------------
Date: 2025-12-3
Steps: 1456
Calories: 245
Workout: 36 minutes
--------------------
Date: 2025-12-3
Steps: 1456
Calories: 245
Workout: 36 minutes
--------------------
```

## Motivation



# Testing Approach

**1. Manual Testing**

The application was tested manually by entering different sets of data and checking the output in the text area.

**2. Input Validation Testing**

- Entered empty fields → ensured no crash
- Entered normal valid inputs → confirmed they were saved correctly
- Entered large values → checked correct file writing

**3. File Handling Testing**

- Tested with no data.txt file → app showed “No data” safely
- Tested with existing data → records were read and displayed correctly
- Verified the 5-line format for each record

**4. View Records Testing**

- Added multiple entries and ensured all were displayed in order
- Checked formatting (Date, Steps, Calories, Workout)

**5. Monthly Records Testing**

- Added more than 30 records → confirmed only last 30 were shown
- Tested with fewer than 30 records → all records displayed correctly

# Challenges Faced

**1. Managing File Formatting**
Using a plain text file required careful handling of the 5-line record structure.
Any small mistake (extra space or missing line) could break the reading loop.

**2. Indexing While Reading Records**
Since each entry uses 5 lines, the loop had to increment correctly (i += 5).
Incorrect indexing caused misalignment of data.

**3. Handling Missing Files**
If data.txt did not exist, the program had to show a message instead of crashing.

**4. Designing a Simple UI Layout**
Arranging Tkinter widgets cleanly and making the interface easy to use required multiple adjustments.

**5. Ensuring Monthly Records Logic Works**
Selecting only the last 30 entries required correct calculation of the starting line (max(0, len(lines) - 150)).

# Learning Outcomes

**1. Understanding of GUI Development**
Learned how to build a functional desktop application using Tkinter, including labels, entry fields, buttons, and text widgets.

**2. File Handling Skills**
Gained experience in reading and writing data using text files, structuring records, and managing file errors safely.

**3. Modular Programming**
Understood how separating features into functions (Add, View, Monthly, Motivation) improves clarity and maintainability.

**4. Logical Thinking & Debugging**
Improved problem-solving skills while handling indexing issues, testing loops, and fixing formatting errors.

**5. User Interface Design**
Learned how to design a simple, clean, and user-friendly UI suitable for beginners.

**6. Real-World Application Development**
Experienced how individual programming concepts combine to create a complete working product.

# Future Enhancements

**1. Add Graphs and Charts**
Include visual trends for steps, calories, and workout minutes using plotting libraries.

**2. Edit and Delete Records**
Allow users to modify or remove previously saved entries.

**3. Switch to Database Storage**
Replace the text file with SQLite for better data management and scalability.

**4. Export Data**
Add an option to export records to CSV or Excel for external use.

**5. Weekly and Monthly Summary**
Provide automatic summaries showing averages and totals.

**6. Improved UI Design**
Use frames, colors, and better layout structure to make the interface more visually appealing.

**7. Add User Authentication**
Optional login system for personalized tracking.

# Conclusion

The Health Tracker project was a simple but meaningful way to bring together everything I learned about Python, Tkinter, and file handling. While building it, I understood how a small idea can turn into a real working application with clear features and a clean interface. The app successfully records daily health information, displays saved data, and provides motivational messages that make the experience more engaging.

Overall, this project helped me improve my problem-solving skills, understand how GUI applications work, and learn how different parts of a program interact with each other. It also showed me how even basic tools can create something useful in everyday life. With more time, this project can be expanded into a more advanced and polished health-tracking application.

# References


- Python Official Documentation
- https://docs.python.org
- Tkinter Documentation
- https://docs.python.org/3/library/tkinter.html
- Random Module Documentation
- https://docs.python.org/3/library/random.html
- Basic Tkinter Tutorials & Examples
- Various online resources (GeeksforGeeks, W3Schools, TutorialsPoint)
- Class Notes / Course Material
- Concepts of Python programming, GUI basics, and file handling from the course syllabus.

# Thank You