# HEALTH TRACKING APP

## (VITYARTHI CSE PROJECT)

NAME: TANISHQA S. DANGE
REG. NO: 25BAI11194
PROGRAMME: CSE AI ML

# Introduction:

This Health Tracking App is a Python-based console application designed to help users monitor their personal health data efficiently. It enables recording and storing of key health metrics such as weight, calorie intake, and exercise duration on a daily basis. The app leverages basic file handling for data persistence and the datetime module to timestamp entries, providing users with an easy way to log and review their health activities over time. Its simple, menu-driven interface makes it accessible for users of all programming levels, offering a practical example of applying fundamental Python concepts to create a meaningful real-world application. This project promotes healthier lifestyle management by encouraging consistent health monitoring in a structured, convenient manner.

## Project Objective Statement

The objective of this Health Tracking App project is to provide students with a hands-on opportunity to address a relevant problem—personal health monitoring—by developing a Python-based solution. This application enables users to record and track daily health metrics such as weight, calorie intake, and exercise routines, leveraging file handling and user input techniques learned in the course.

## Real-World Problem Identification

Many individuals struggle to consistently monitor and evaluate their daily health parameters. Without systematic tracking, it's difficult to observe progress, set goals, or make informed decisions about diet and exercise. This project addresses the challenge by providing a simple system for logging and viewing weight, calories, and exercise data in a user-friendly way.

## Technical Solution Design

- The solution employs Python's built-in file handling capabilities to store health records in text files.
- It uses the `datetime` module to timestamp each entry, ensuring data relevance and traceability.
- Interactive functions capture user input for weight, calories, and exercise, while a menu-driven loop in the main function supports continual use.
- The app also allows retrieval and display of past data, making personal health review accessible and organized.

This project demonstrates a clear understanding of core Python programming, problem-solving, real-world application design, and documentation. By walking users through data entry, record storage, and review, it validates mastery of:

- File and input handling
- Loop and control flow structures
- Modular code organization
- Simple error detection for user-friendly experience

This approach aligns well with the academic goals of applying theoretical knowledge to realistic scenarios, reinforcing practical skill development in programming and technical documentation.

## Functional Requirements:

- Three major functional modules: (1) Record weight, (2) Record calories consumed, (3) Record exercise details.
- Clear input/output structure: User inputs data via console prompts; data stored in text files and can be viewed on demand.
- Logical workflow: Menu-driven system lets users select tasks, input data, view logs, or exit, ensuring smooth interaction.

## Non-Functional Requirements:

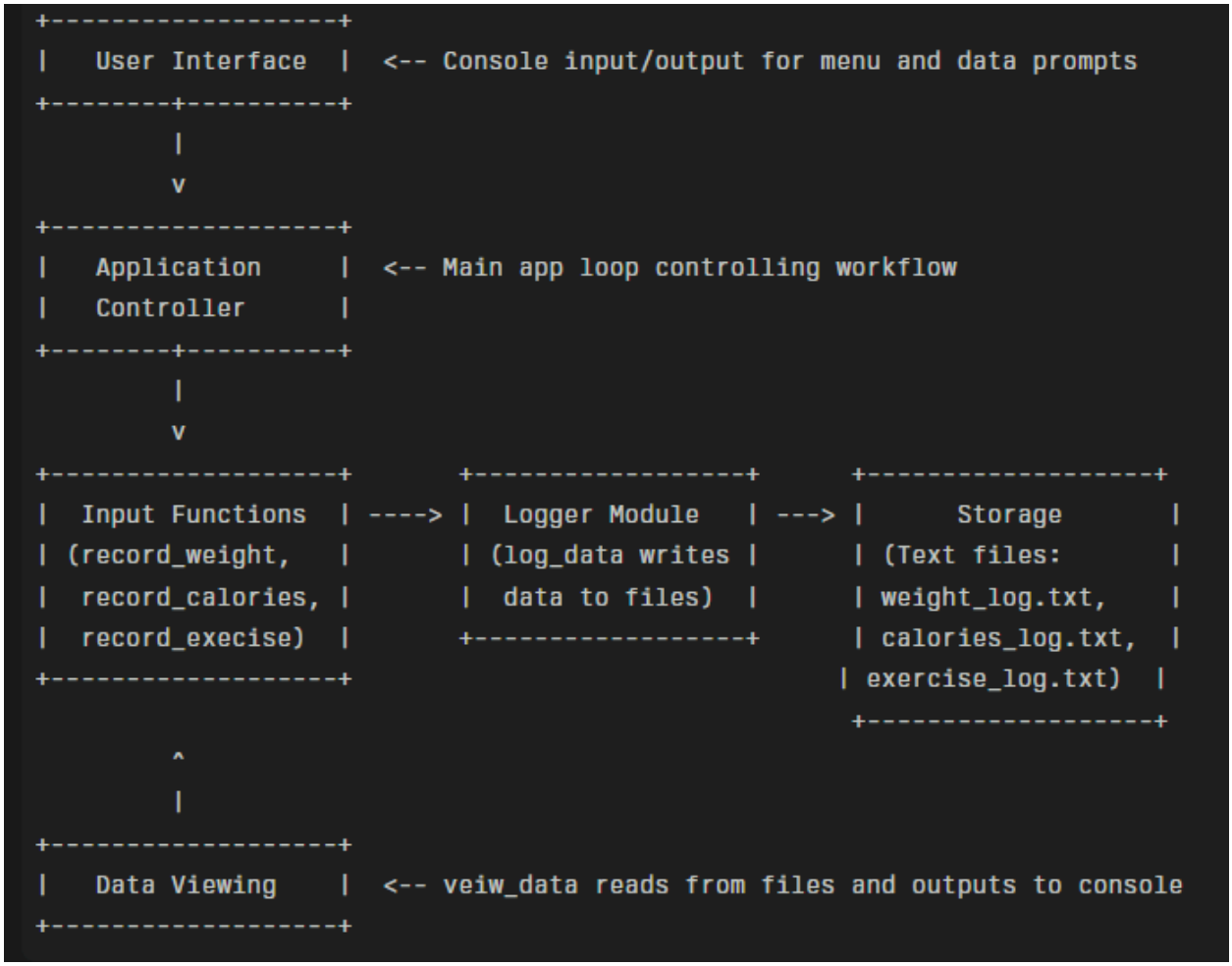
- Usability: Simple text-based menu and prompts make the app easy to use.
- Reliability: Exception handling for missing files during data viewing ensures stable operation.
- Maintainability: Modular function design allows easy updates and debugging.
- Error handling strategy: User choices validated with feedback for invalid input, preventing crashes.

This project captures essential health data through straightforward user interaction, backed by robust file storage and user-friendly prompts, meeting the scope and requirements effectively.

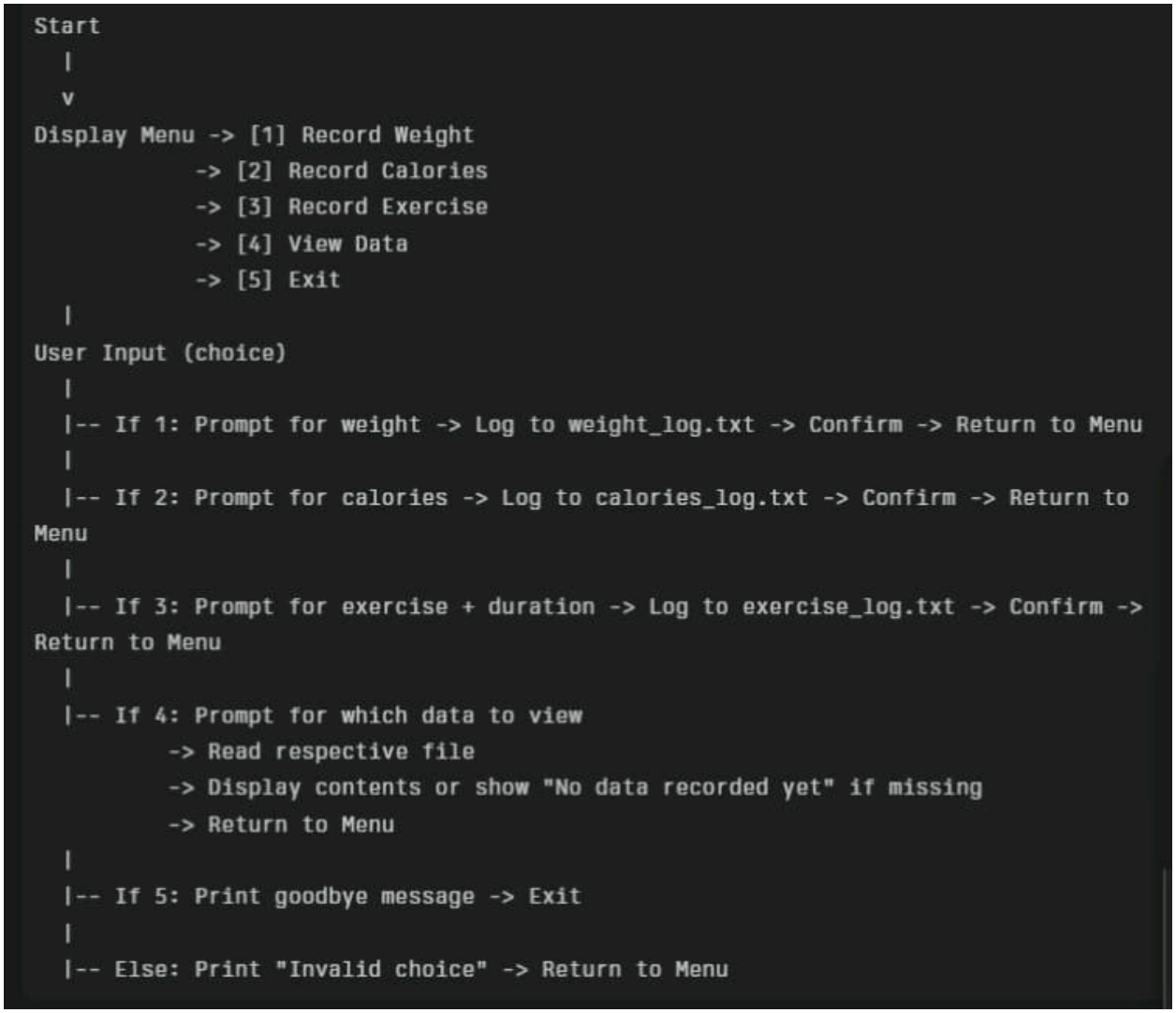
## Technical Expectations based on Health Tracking App code:

- Architectural design: The app follows a modular, function-based structure separating concerns such as logging, data recording, viewing, and user interaction. The main loop controls the flow logically.
- Application of concepts: Uses file handling for persistent data, datetime module for date stamping, and input/output for user interaction, demonstrating core Python programming principles.
- Modular and clean: Functions like log_data, record_weight, record_calories, record_execise, and veiw_data isolate specific functionalities, improving readability and maintainability.
- Documentation and comments: Code includes basic comments explaining the purpose of functions and major sections, which aids understanding.
- Validation and error handling: Handles file-not-found errors when viewing data and checks invalid menu choices to guide users properly

## System Architecture Diagram:

- 


## Process Flow or Workflow Diagram:




# UML diagrams based on Health Tracking App code:

### Use Case Diagram:

- Actors: User
- Use Cases:
    - Record weight
    - Record calories consumed
    - Record exercise details
    - View recorded data
    - Exit application

- The user interacts with the system to perform these functions via console inputs.

**Component Diagram:**

- Components:
    - User Interface (Console input/output)
    - Data Entry Modules (record_weight, record_calories, record_execise)
    - Data Logger (log_data function writing to files)
    - Data Storage (weight_log.txt, calories_log.txt, exercise_log.txt files)
    - Data Viewer (veiw_data function reading from files)
- The User Interface connects to Data Entry Modules and Data Viewer, which in turn interact with Data Logger and Storage.

**Sequence Diagram (example for recording weight):**

1. User selects "Record weight" from the menu
2. System prompts user to enter weight
3. User inputs weight
4. System calls record_weight()
5. record_weight calls log_data() with weight and current date
6. log_data appends data to weight_log.txt
7. System prints confirmation message
8. Returns to main menu

# Testing Approach:

- Manual testing was performed by entering various valid and invalid inputs through the console for weight, calories, and exercise.
- Verified correct logging of input data into respective text files with proper date stamping.
- Tested viewing function for handling existing records and graceful error handling when files are missing or empty.
- Checked menu navigation and invalid choice handling to ensure robust user experience.

# Challenges Faced:

- Correctly managing date stamping in log entries (noticing a typo in `datetime.data.today()` which should be `datetime.date.today()`).
- Handling user input validation to prevent crashes or incorrect records (basic validation implemented; can be improved).

- Ensuring file I/O operations handle missing or malformed files without crashing.
- Balancing simplicity of a console app with usability and meaningful feedback.

## Learnings & Key Takeaways:

- Importance of modular programming for clarity and maintainability in coding projects.
- How to implement effective file handling in Python for persistent data storage.
- Practical use of the datetime module to timestamp records.
- Value of clear user communication through input prompts and error messages.
- The necessity of handling exceptions to improve application stability.

## Future Enhancements:

- Add input validation to ensure numeric inputs for weight, calories, and duration.
- Introduce data editing and deletion (CRUD operations) for existing records.
- Implement summary reports or analytics (e.g. trends over time).
- Develop a graphical user interface (GUI) for enhanced usability.
- Store data in a structured database instead of text files for scalability.
- Add user authentication for privacy and multi-user support.

## References:


- Python official documentation: file handling and datetime modules.
- Programming tutorials and course materials on Python programming and basic software design principles.
- Examples of basic health tracking apps in Python from open-source repositories and educational sources.