System Requirements Specification Index

For

Nutrition Calculator System

Version 1.0

IIHT Pvt. Ltd.
fullstack@iiht.com

TABLE OF CONTENTS

Nutrition Calculator System
System Requirements Specification

1 Project Abstract

The Health Monitoring System (HMS) requires a set of basic nutrition calculation functions to process user dietary information. This assignment focuses on implementing and calling functions that perform simple nutritional calculations. The emphasis is on proper function definition, documentation, and invocation rather than complex nutritional algorithms.

2 Business Requirements:

Screen Name	Console input screen
Problem Statement	1. System needs properly defined functions with appropriate parameters 2. Each function must be properly documented with docstrings 3. Functions must be called with correct arguments 4. System must demonstrate basic nutrition calculations like Calorie calculation, Macronutrient calculation, Body mass index calculation, Water intake recommendation

3 Constraints

3.1 Input Requirements

1. BMI Calculation:
    - Weight must be in kilograms (positive number)
    - Height must be in meters (positive number)
    - Function must validate input types and values

2. BMI Category:
    - Categories must be defined as: "Underweight" (< 18.5), "Normal weight" (18.5-24.9), "Overweight" (25-29.9), "Obese" (≥ 30)
    - Function must validate input type
3. Calorie Calculation:
    - Activity levels must be one of: "sedentary", "light", "moderate", "active", "very active"
    - Gender must be either "male" or "female"
    - Age must be a positive integer
    - Function must validate all inputs
4. Protein Requirements:
    - Activity levels must be one of: "light", "moderate", "intense"
    - Function must use protein factors: light (1.2g/kg), moderate (1.6g/kg), intense (2.0g/kg)
5. Water Intake:
    - Activity factor must be 1.0 or greater
    - Base calculation uses 33.3ml per kg of body weight

3.2 FUNCTION DEFINITION REQUIREMENTS

1. Function Structure:
    - Each function must have a clear name indicating its purpose
    - Parameters must have descriptive names
    - Functions must use return statements appropriately
    - Example: `def calculate_bmi(weight_kg, height_m):`
2. Docstrings:
    - Each function must include a docstring describing:
        - Purpose of the function
        - Parameters (name, type, description)
        - Return value (type, description)
        - Example usage
3. Parameter Types:
    - Functions must accept basic data types (int, float, str, list)

- Some functions should accept multiple parameters
- At least one function should have a default parameter value

4. Return Values:

- Functions must return appropriate data types
- Some functions must return numeric values
- At least one function should return a formatted string

3.3 Operations Constraints

1. Basic Calling:
    - Functions must be called with positional arguments
    - Example: `result = calculate_bmi(75, 1.8)`
2. Keyword Arguments:
    - Some function calls must use keyword arguments
    - Example: `result = calculate_calories(weight_kg=70, activity_level="moderate")`
3. Default Parameters:
    - Functions with default parameters must be called both with and without providing the default parameter
    - Example: `water_intake = calculate_water_intake(70)` and `water_intake = calculate_water_intake(70, activity_factor=1.5)`
4. Function Call Composition:
    - At least one example must show calling a function and using its return value as an argument for another function
    - Example: `bmi_category = get_bmi_category(calculate_bmi(weight, height))`
5. Return Values:
    - Functions must return consistent data types
    - Validation functions must return validation result
    - Example: `return {"is_valid": True, "message": "Food item data is valid"}`

3.4 Output Constraints

1. Display Format:

- Main function must display results of function calls in a readable format
- Example: `print(f"BMI: {bmi:.1f} - Category: {category}")`

- appropriately

3. Output Format:
    - Console output must follow this format:
        - Show "== NUTRITION CALCULATOR=="
        - Show each function call result in clearly labeled sections
        - Format numeric outputs with appropriate decimal places
        - Include appropriate units in the output (kg, cm, kcal, etc.)

4. TEMPLATE CODE STRUCTURE:

1. Basic Calculation Functions:

- `calculate_bmi(weight_kg, height_m)` - calculates body mass index
- `get_bmi_category(bmi)` - returns BMI category (underweight, normal, overweight, etc.)
- `calculate_calories(weight_kg, height_m, age, gender, activity_level)` - calculates daily calorie needs

2. Nutrient Functions:

- `calculate_protein_needs(weight_kg, activity_level="moderate")` - calculates protein needs with default activity level
- `calculate_water_intake(weight_kg, activity_factor=1.0)` - calculates water intake with default activity factor

3. Helper Functions:

- `format_nutrition_result(calories, protein, carbs, fat)` - formats nutrition information for display

4. Main Program Function:

- `main()` - main program function demonstrating all other function definitions and calls
- Must demonstrate basic positional arguments
- Must demonstrate keyword arguments
- Must demonstrate default parameter usage
- Must demonstrate function composition
- Must format and display all results

5. EXECUTION STEPS TO FOLLOW:

1. Run the program
2. Observe the definition of each function
3. See examples of function calls with different parameter passing methods
4. View results of basic nutrition calculations

5. Observe how function outputs are used in other function calls
6. See formatted output of calculations