System Requirements Specification Index

For

Text Processing System

Version 1.0

IIHT Pvt. Ltd.
fullstack@iiht.com

TABLE OF CONTENTS

1 Project Abstract

TextMaster Systems requires a comprehensive string processing application to manipulate, analyze, and format text data. The system will perform character-level operations, content extraction, transformation, and formatting functions on various text sources. This tool will enable users to work with plain text, structured data formats, and semi-structured content while demonstrating core string manipulation capabilities.

2 Business Requirements:

Screen Name	Console input screen
Problem Statement	1. System needs to process different text formats (plain text, formatted text, code, CSV, JSON) 2. System must support basic text analysis operations (character counting, word counting, pattern detection) 3. Console should handle different string operations like String slicing and indexing (accessing specific parts of text), String methods (upper/lower case conversion, strip, join, split), String analysis (palindrome detection, vowel/consonant counting), Pattern extraction (email addresses, dates, urls), Text formatting and tabulation

3 Constraints

3.1 Input Requirements

1. Food Item Data:

- Each food item record must include id, name, category, quantity, unit, expiration_date, storage_location
- Example: `{"id": "F001", "name": "Apples", "category": "Produce", "quantity": 25, "unit": "kg", "expiration_date": "2023-12-15", "storage_location": "Cooler 3"}`

2. Donation Opportunity Data:
   - Each donation opportunity must include id, name, accepts_categories
   - Example: `{"id": "R001", "name": "City Food Bank", "accepts_categories": ["Produce", "Canned", "Bakery"]}`
3. Food Categories:
   - Must be one of: "Produce", "Dairy", "Bakery", "Meat", "Frozen", "Canned", "Dry Goods", "Prepared"
4. Variable Names:
   - Food inventory must be stored in a variable named `inventory`
   - Recipients must be stored in a variable named `recipients`

3.2 Operations Constraints

1. **Function Definition:**
   - Each function must have a clear purpose and responsibility
   - Must use proper parameter naming
   - Must include return value documentation
   - Example: `def validate_food_item(food_item):`
2. **Docstrings:**
   - Each function must include a docstring describing:
     - Purpose of the function
     - Parameters (name, type, description)
     - Return value (type, description)
     - Example usage
3. **Error Handling:**
   - Functions must handle invalid inputs appropriately
   - Must return appropriate error messages or raise exceptions
   - Example: `if not isinstance(quantity, (int, float)) or quantity < 0: raise ValueError("Quantity must be a positive number")`
4. **Parameter Validation:**

- Functions must validate parameter types and values
- Example: `if not isinstance(food_id, str): raise TypeError("Food ID must be a string")`

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"}`

3.3 Output Constraints

1. Display Format:
    - Functions must return properly formatted data
    - Display functions must format output appropriately
    - Example: `return f"{food_item['name']} - {food_item['quantity']} {food_item['unit']} - Expires: {food_item['expiration_date']}"`
2. Output Format:
    - Console output must follow this format:
        - Show "== FOOD WASTE REDUCTION SYSTEM =="
        - Show function results in clearly labeled sections
        - Format quantities with appropriate units
        - Display dates in YYYY-MM-DD format

4. Template code Structure:

1. Data Validation Functions:
    - `validate_food_item(food_item)` - validates food item data structure
    - `calculate_days_until_expiration(expiration_date)` - calculates days until expiration

2. Data Processing Functions:
    - `identify_expiring_items(food_items, days_threshold)` - identifies items expiring soon
    - `sort_items_by_expiration(food_items)` - sorts items by expiration date
    - `match_donations(food_items, recipients)` - matches food items with recipients

3. Display Functions:
    - `format_food_item(food_item)` - formats food item information for display

4. Program Control Functions:
   - `main()` - main program function demonstrating all other functions

5. EXECUTION STEPS TO FOLLOW:

1. Run the program
2. Observe the execution of each function
3. View validation results
4. See expiration date calculations
5. View expiring items identification
6. Observe donation matching algorithm
7. See formatted output