System Requirements Specification Index

For

Online Store Management Console Application

Version 1.0

IIHT Pvt. Ltd.
fullstack@iiht.com

TABLE OF CONTENTS

Online Store Management System

System Requirements Specification

1 PROJECT ABSTRACT

TechBazaar, an online electronics retailer, requires an inventory management system to organize its growing product catalog. The system will track products, filter inventory, update prices, and generate reports. It will enable store managers to efficiently manage inventory by categorizing products, filtering by various criteria, updating inventory data, and calculating statistics. This system provides an efficient way for staff to organize and analyze their product catalog.

2 BUSINESS REQUIREMENTS:

Screen Name	Console input screen
Problem Statement	1. System needs to store and categorize different types of products (electronics, clothing, groceries, footwear, health) 2. System must support filtering products by category, price range, availability, or feature 3. Program must apply various transformations to lists (reverse, slice, extend, shuffle) 4. Console should handle different dictionary operations like Basic filtering (products by category, availability), Dictionary comprehension (for price ranges, features), Dictionary methods (update, get, items), Dictionary merging and transformation, Dictionary-based data analysis

3 CONSTRAINTS

3.1 INPUT REQUIREMENTS

1. Product Records:

- Must be stored as dictionaries with fields for id, name, category, price, stock, rating, features
- Must be stored in a dictionary variable with product ID as key
- Example: `"P001": {"name": "Smartphone XS", "category": "electronics", "price": 59999.99, "stock": 25, "rating": 4.5, "features": ["5G", "128GB Storage", "Dual Camera"]}`

2. Product Categories
   - Must be one of: "electronics", "clothing", "groceries", "footwear", "health"
   - Must be stored as string in product's "category" field
   - Example: "electronics"
3. Availability Status:
   - Must be determined by "stock" field value
   - Example: Product is available if stock > 0
4. Rating Score:
   - Must be stored as float in "rating" field
   - Must be between 1.0-5.0
   - Example: 4.5
5. Predefined Products:
   - Must use these exact predefined products in the initial product dictionary:
     - `"P001": {"name": "Smartphone XS", "category": "electronics", "price": 59999.99, "stock": 25, "rating": 4.5, "features": ["5G", "128GB Storage", "Dual Camera"]}`
     - `"P002": {"name": "Designer Jeans", "category": "clothing", "price": 4999.99, "stock": 40, "rating": 4.2, "features": ["Slim Fit", "Stretch Denim", "Dark Wash"]}`
     - `"P003": {"name": "Bluetooth Headphones", "category": "electronics", "price": 7999.99, "stock": 15, "rating": 4.7, "features": ["Noise Cancelling", "40hr Battery", "Hi-Fi Sound"]}`
     - `"P004": {"name": "Organic Coffee Beans", "category": "groceries", "price": 899.99, "stock": 50, "rating": 4.8, "features": ["Fair Trade", "Whole Bean", "Medium Roast"]}`

- `"P005": {"name": "Running Shoes", "category": "footwear", "price": 6999.99, "stock": 30, "rating": 4.6, "features": ["Breathable", "Cushioned", "Lightweight"]}`

6. New Products:
    - Must use these exact predefined items in the new products dictionary:
        - `"N001": {"name": "Smart Watch", "category": "electronics", "price": 15999.99, "stock": 20, "rating": 4.4, "features": ["Heart Rate Monitor", "GPS", "Water Resistant"]}`
        - `"N002": {"name": "Protein Powder", "category": "health", "price": 1999.99, "stock": 45, "rating": 4.3, "features": ["Plant-Based", "20g Protein", "Sugar-Free"]}`

3.2 OPERATIONS CONSTRAINTS

1. Dictionary Creation:
    - Must use proper dictionary creation syntax
    - Example: `{"id": "P001", "name": "Product Name", ...}`

2. Dictionary Access:
    - Must use proper key access methods
    - Example: `inventory[product_id]` or `inventory.get(product_id)`
3. Dictionary Filtering:
    - Must use dictionary comprehension
    - Example: `{pid: product for pid, product in inventory.items() if product["category"] == "electronics"}`

4. Dictionary Merging:
    - Must use dictionary unpacking
    - Example: `{**existing_dict, **new_dict}`

5. Dictionary Transformation:
    - Must use dictionary unpacking with modification
    - Example: `{**product, "new_arrival": True}`

6. Dictionary Methods:
    - Must use dictionary methods (keys, values, items)
    - Example: `inventory.items()`, `inventory.keys()`

7. Dictionary-based Analytics:
    - Must use dictionaries for storing and calculating statistics
    - Example: `category_counts = {}`

8. Error Handling:
    - Must check if keys exist before accessing
    - Example: `if product_id in inventory:`

9. Immutability:
    - Must create new dictionaries rather than modifying in place
    - Example: `updated_inventory = inventory.copy()`

10. Dictionary Comprehension:
    - Must use dictionary comprehension for filtering and transforming
    - Example: `{k: v for k, v in d.items() if condition}`

3.3 OUTPUT CONSTRAINTS

1. Dictionary Format:

- Show product ID, name, category, price, stock, rating, features
- Format price with ₹ symbol
- Format rating with star rating (★)
- Each product must be displayed on a new line

2. Required Output Format:

- Must show in this order:
    - Show "== ONLINE STORE MANAGEMENT SYSTEM =="
    - Show "Total Products: {count}"
    - Show "Categories: {categories}"
    - Show "Current Inventory:"
    - Show products with format: "{id} | {name} | {category} | ₹{price} | Stock: {stock} | Rating: {stars} | {features}"
    - Show "Filtered Results:" when displaying search results

4. TEMPLATE CODE STRUCTURE:

1. Data Management Functions:

- `initialize_data()` - creates the initial inventory and new products dictionaries

2. Dictionary Operation Functions:

- `filter_by_category(inventory, category)` - filters products by category
- `filter_by_price_range(inventory, min_price, max_price)` - filters products by price range
- `filter_by_availability(inventory, min_stock)` - filters products by availability
- `filter_by_feature(inventory, feature)` - filters products by feature
- `find_products_with_keyword(inventory, keyword)` - finds products with keyword

- `update_product_price(inventory, product_id, new_price)` - updates a product's price
- `update_stock_level(inventory, product_id, quantity_change)` - updates stock level
- `add_product_feature(inventory, product_id, new_feature)` - adds a feature to a product
- `merge_inventories(existing_inventory, new_products)` - merges two inventory dictionaries
- `calculate_category_counts(inventory)` - counts products in each category
- `calculate_total_inventory_value(inventory)` - calculates total inventory value
- `find_highest_rated_product(inventory)` - finds highest rated product
- `create_price_brackets(inventory)` - groups products into price brackets

3. Display Functions:

- `get_formatted_product(pid, product)` - formats a product for display
- `display_data(data, data_type)` - displays products or other data types

4. Program Control Functions:

- `main()` - main program function

5. EXECUTION STEPS TO FOLLOW:

1. Run the program
2. View the main menu
3. Select operations:
   - Option 1: View Inventory
   - Option 2: Filter Products
   - Option 3: Update Products
   - Option 4: Add New Products
   - Option 5: View Statistics
   - Option 0: Exit
4. Perform operations on the product inventory
5. View results after each operation
6. Exit program when finished