System Requirements Specification Index

For

Wildlife Conservation Tracking System

Version 1.0

IIHT Pvt. Ltd.
fullstack@iiht.com

TABLE OF CONTENTS

1 PROJECT ABSTRACT

The Indian Wildlife Foundation (IWF) requires a specialized tracking system to monitor endangered species across various wildlife sanctuaries in India. The system will track animal populations, monitor habitat conditions, record conservation efforts, and generate analytical reports. It will enable conservation officers to efficiently manage wildlife data by categorizing species by conservation status, filtering by habitat type, updating population counts, and calculating conservation metrics. This system provides an efficient way for wildlife experts to organize and analyze critical conservation data.

2 BUSINESS REQUIREMENTS:

Screen Name	Console input screen
Problem Statement	1. System needs to store and categorize different types of wildlife data (species, population, habitat, conservation status) 2. System must support filtering species by conservation status, habitat type, population trend, or specific sanctuary 3. Console should handle different dictionary operations like Basic filtering (species by conservation status, habitat type), Dictionary comprehension (for population ranges, habitat conditions), Dictionary methods (update, get, items), Dictionary merging and transformation, Dictionary-based data analysis

3 Constraints

3.1 Input Requirements

1. Species Records:
   - Must be stored as dictionaries with fields for id, name, scientific_name, conservation_status, population, habitat_type, sanctuaries, threats
   - Must be stored in a dictionary variable with species ID as key
   - Example: `"SP001": {"name": "Bengal Tiger", "scientific_name": "Panthera tigris tigris", "conservation_status": "Endangered", "population": 3500, "habitat_type": "Forest", "sanctuaries": ["Sundarbans", "Jim Corbett", "Bandhavgarh"], "threats": ["Poaching", "Habitat Loss", "Human Conflict"]}`
2. Conservation Status:
   - Must be one of: "Least Concern", "Near Threatened", "Vulnerable", "Endangered", "Critically Endangered"
   - Must be stored as string in species' "conservation_status" field
   - Example: "Endangered"
3. Habitat Types:
   - Must be one of: "Forest", "Grassland", "Wetland", "Mountain", "Desert"
   - Must be stored as string in species' "habitat_type" field
   - Example: "Forest"
4. Population:
   - Must be stored as integer in "population" field
   - Must be greater than or equal to 0
   - Example: 3500
5. Predefined Species:
   - Must use these exact predefined species in the initial species dictionary:
     - `"SP001": {"name": "Bengal Tiger", "scientific_name": "Panthera tigris tigris", "conservation_status": "Endangered", "population": 3500, "habitat_type": "Forest", "sanctuaries": ["Sundarbans", "Jim Corbett", "Bandhavgarh"], "threats": ["Poaching", "Habitat Loss", "Human Conflict"]}`
     - `"SP002": {"name": "Asian Elephant", "scientific_name": "Elephas maximus", "conservation_status": "Endangered", "population": 27000, "habitat_type": "Forest", "sanctuaries": ["Periyar", "Nagarhole", "Jim Corbett"], "threats": ["Habitat Loss", "Human Conflict", "Poaching"]}`

        - `"SP003": {"name": "Indian Rhinoceros", "scientific_name": "Rhinoceros unicornis", "conservation_status": "Vulnerable", "population": 3600, "habitat_type": "Grassland", "sanctuaries": ["Kaziranga", "Manas", "Orang"], "threats": ["Poaching", "Habitat Loss", "Flooding"]}`

        - `"SP004": {"name": "Snow Leopard", "scientific_name": "Panthera uncia", "conservation_status": "Vulnerable", "population": 450, "habitat_type": "Mountain", "sanctuaries": ["Hemis", "Pin Valley", "Great Himalayan"], "threats": ["Climate Change", "Poaching", "Prey Depletion"]}`

        - `"SP005": {"name": "Indian Vulture", "scientific_name": "Gyps indicus", "conservation_status": "Critically Endangered", "population": 30000, "habitat_type": "Grassland", "sanctuaries": ["Ranthambore", "Pench", "Bandhavgarh"], "threats": ["Diclofenac Poisoning", "Habitat Loss", "Food Scarcity"]}`

6. New Species:

    - Must use these exact predefined items in the new species dictionary:

        - `"NS001": {"name": "Ganges River Dolphin", "scientific_name": "Platanista gangetica", "conservation_status": "Endangered", "population": 3500, "habitat_type": "Wetland", "sanctuaries": ["Vikramshila", "National Chambal", "Katerniaghat"], "threats": ["Water Pollution", "Fishing Nets", "Dams"]}`

        - `"NS002": {"name": "Great Indian Bustard", "scientific_name": "Ardeotis nigriceps", "conservation_status": "Critically Endangered", "population": 150, "habitat_type": "Grassland", "sanctuaries": ["Desert National Park", "Kutch Bustard", "Rollapadu"], "threats": ["Habitat Loss", "Power Lines", "Predation"]}`

3.2 Operations Constraints

1. Dictionary Creation:
    - Must use proper dictionary creation syntax
    - Example: `{"id": "SP001", "name": "Species Name", ...}`
2. Dictionary Access:
    - Must use proper key access methods
    - Example: `species_data[species_id]` or `species_data.get(species_id)`
3. Dictionary Filtering:
    - Must use dictionary comprehension
    - Example: `{sid: species for sid, species in species_data.items() if species["conservation_status"] == "Endangered"}`
4. Dictionary Merging:

- Must use dictionary unpacking
- Example: `{**existing_dict, **new_dict}`

5. Dictionary Transformation:
   - Must use dictionary unpacking with modification
   - Example: `{**species, "newly_added": True}`
6. Dictionary Methods:
   - Must use dictionary methods (keys, values, items)
   - Example: `species_data.items()`, `species_data.keys()`
7. Dictionary-based Analytics:
   - Must use dictionaries for storing and calculating statistics
   - Example: `status_counts = {}`
8. Error Handling:
   - Must check if keys exist before accessing
   - Example: `if species_id in species_data:`
9. Immutability:
   - Must create new dictionaries rather than modifying in place
   - Example: `updated_species_data = species_data.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. Display Format:
   - Show species ID, name, scientific name, conservation status, population, habitat type, sanctuaries, threats
     - Format population with thousands separator
     - Format conservation status with color indicators (optional)
     - Each species must be displayed on a new line
2. Output Format:
   - Must show in this order:
     - Show "== WILDLIFE CONSERVATION TRACKING SYSTEM =="
     - Show "Total Species: {count}"

- Show "Conservation Statuses: {statuses}"
- Show "Current Species Data:"
- Show species with format: "{id} | {name} ({scientific_name}) | {conservation_status} | Population: {population} | Habitat: {habitat_type} | Sanctuaries: {sanctuaries} | Threats: {threats}"
- Show "Filtered Results:" when displaying search results

4. TEMPLATE CODE STRUCTURE:

1. Data Management Functions:

- `initialize_data()` - creates the initial species and new species dictionaries

2. Dictionary Operation Functions:

- `filter_by_conservation_status(species_data, status)` - filters species by conservation status
- `filter_by_population_range(species_data, min_population, max_population)` filters species by population range
- `filter_by_habitat_type(species_data, habitat_type)` - filters species by habitat type
- `filter_by_sanctuary(species_data, sanctuary)` - filters species by sanctuary
- `find_species_with_keyword(species_data, keyword)` - finds species with keyword
- `update_species_population(species_data, species_id, new_population)` - updates a species' population
- `update_conservation_status(species_data, species_id, new_status)` - updates conservation status
- `add_species_threat(species_data, species_id, new_threat)` - adds a threat to a species
- `merge_species_data(existing_species, new_species)` - merges two species dictionaries
- `calculate_status_counts(species_data)` - counts species in each conservation status
- `calculate_total_population(species_data)` - calculates total population across all species
- `find_most_threatened_species(species_data)` - finds most threatened species
- `create_population_brackets(species_data)` - groups species into population brackets

3. Display Functions:

- `get_formatted_species(sid, species)` - formats a species for display
- `display_data(data, data_type)` - displays species or other data types

4. Program Control Functions:
- display_menu() -> None
- `main()` - main program function

5. EXECUTION STEPS TO FOLLOW:

1. Run the program
2. View the main menu
3. Select operations:
   - Option 1: View Species Data
   - Option 2: Filter Species
   - Option 3: Update Species Data
   - Option 4: Add New Species
   - Option 5: View Conservation Statistics
   - Option 0: Exit
4. Perform operations on the species data
5. View results after each operation
6. Exit program when finished