System Requirements Specification Index

For

Date and Time Processor

Version 1.0

IIHT Pvt. Ltd.
fullstack@iiht.com

TABLE OF CONTENTS

Date and Time Processor
System Requirements Specification

1 Project Abstract

A healthcare scheduling system needs reliable date and time processing capabilities to manage patient appointments, medication schedules, and staff rotations. This assignment focuses on implementing fundamental datetime operations using Python's datetime module to create a practical scheduling tool. Students will learn how to perform essential datetime manipulations including conversions, calculations, formatting, and timezone handling for real-world scheduling scenarios.

2 Business Requirements:

Screen Name	Console input screen
Problem Statement	1. System must implement core datetime manipulation functions 2. Functions must handle date arithmetic, formatting, and timezone conversions 3. All functions require proper documentation (docstrings) 4. System must demonstrate practical applications of datetime processing 5. Error handling must validate inputs and handle edge cases

3 Constraints

3.1 Input Requirements

1. Date and Time Formats:
    - String dates must follow ISO format (YYYY-MM-DD)

- String datetimes must follow format (YYYY-MM-DD HH:MM:SS)
- Example: `"2025-03-19 14:30:00"`

2. Function Parameters:
   - Date strings must be validated before processing
   - Timezone values must be integer offsets (hours from UTC)
   - Duration inputs must be integers for days, hours, minutes
   - Date ranges must have valid start and end dates

3.2 FUNCTION CONSTRAINTS

1. Function Definition:
   - Each function must have a specific datetime manipulation purpose
   - Must include docstrings with examples
   - Example: `def convert_string_to_datetime(date_string):`

2. Datetime Operations:
   - Must use Python's `datetime` module
   - Must handle string parsing and formatting
     - `convert_string_to_datetime()`: Parse string to datetime object
     - `format_datetime()`: Format datetime to specified string pattern
     - `calculate_date_difference()`: Find days between two dates
     - `add_time_duration()`: Add specified time to a datetime
     - `get_day_of_week()`: Return weekday name for given date
     - `convert_timezone()`: Convert datetime between timezone offsets

3. Return Values:
   - Functions must return appropriate types (datetime objects, strings, integers)
   - Functions should return None or empty results when inputs are invalid
   - The calculate_date_difference function must return a dictionary with different time units

4. Error Handling:
   - Functions should validate input types and formats
   - Handle invalid dates, times, and timezone offsets
   - Raise appropriate exceptions with clear error messages

3.3 Output Constraints

1. Display Format:
    - Console output must be clearly labeled
    - Example: `Time difference: 7 days, 5 hours, 30 minutes`
    - Results should be formatted for readability
    - Main function should demonstrate each datetime operation with labeled results

4. Template code Structure:

1. Datetime Conversion Functions:

- `convert_string_to_datetime(date_string)` - converts string to datetime object
- `format_datetime(dt, format_string)` - formats datetime to specified string format

2. Datetime Calculation Functions:

- `calculate_date_difference(start_date, end_date)` - calculates time between dates
- `add_time_duration(dt, days=0, hours=0, minutes=0)` - adds time to datetime
- `get_day_of_week(date_string)` - returns weekday name for date

3. Timezone Functions:

- `convert_timezone(dt, source_offset, target_offset)` - converts datetime between timezone offsets

4. Program Control:

- `main()` - demonstrates all datetime functions with sample inputs
- Should display results clearly with appropriate formatting

5. Execution Steps to follow:

1. Run the program
2. Observe how each datetime function processes date and time data
3. Test with sample inputs for string parsing, calculations, and formatting
4. Experiment with different datetime operations for scheduling scenarios