System Requirements Specification Index

For

Legal String Processor System

Version 1.0

IIHT Pvt. Ltd.
fullstack@iiht.com

TABLE OF CONTENTS

1 PROJECT ABSTRACT

The Legal String Processor is a Python application designed to handle and manipulate text data commonly found in legal documents. It provides functionality for extracting specific information from case citations, client records, contracts, and legal agreements.

2 BUSINESS REQUIREMENTS:

Screen Name	Console input screen
Problem Statement	1. Extract case names and party information from legal citations 2. Process client information from standardized formats 3. Locate and extract specific sections from legal documents 4. Identify and extract dates in YYYY-MM-DD format 5. Perform basic string operations on legal text

3 CONSTRAINTS

3.1 INPUT REQUIREMENTS

1. Text Data Structure:
    - Case citations in standard format (e.g., "Smith v. Jones, 123 F.3d 456 (9th Cir. 2023)")
    - Client information with labeled fields (e.g., "Client:", "DOB:", "Case #:")
    - Structured legal documents with section headings
    - Documents containing dates in YYYY-MM-DD format

3.2 Operations Constraints

- Must handle various citation formats (v. or vs.)
- Must extract information without modifying original text
- Must handle edge cases (missing fields, improper formatting)
- Must provide proper error handling for invalid inputs

3.3 Output Constraints

1. Display Format:
   - Show original text and operation performed
   - Present extracted information clearly
   - Indicate when requested information is not found

1. Output Format:
   - Show original text and operation performed
   - Present extracted information clearly
   - Indicate when requested information is not found

4. Template code Structure:

1. Core Functions:

- `initialize_legal_samples()`: Provides sample legal text data
- `extract_case_name(citation)`: Extracts case name from a citation
- `extract_parties(case_name)`: Separates plaintiff and defendant
- `extract_date(text)`: Finds dates in YYYY-MM-DD format
- `find_section(document, heading)`: Locates document sections
- `extract_client_info(text, field)`: Gets specific client data
- `display_result(original, operation, result)`: Formats output
- `main()`: Handles user interaction and program flow

3. Error Handling:

- Input validation for all functions
- Graceful handling of missing or malformed data
- Clear error messages for invalid operations

4. Program Control Functions:

- `main()` - main program function

5. EXECUTION STEPS TO FOLLOW:

1. Run the program
2. Choose from available sample texts (case citation, client info, etc.)
3. Select an operation to perform:
   - Extract case name and parties
   - Extract client information
   - Find document section
   - Extract date
   - Perform basic string operations
4. View the results of the operation
5. Continue with additional operations or exit