Requirement Analysis

1. Introduction

The purpose of this project is to automate the process of extracting claim dates from a large dataset of URLs. The dataset is provided in a CSV file named `FACTors.csv`, which contains over 118,000 URLs. The extraction process involves scraping each URL, identifying claim date information from its HTML content, and storing the results in a structured CSV format.

2. Functional Requirements

The system must:

1. Accept an input file (`input_urls.csv`) containing URLs to be processed.
2. For each URL, fetch the web page content using HTTP requests.
3. Parse the HTML content to locate and extract claim date information using predefined patterns or parsing logic.
4. Handle failed extractions gracefully, recording URLs without claim dates for further review.
5. Save the final results, including URLs, claim dates, and extraction statuses, in an output CSV file.
6. Provide logging to track progress, errors, and warnings.
7. Support resuming scraping for only failed URLs to save time.

3. Non-Functional Requirements

1. **Performance**:
    - The program should efficiently handle large datasets (over 100,000 URLs).
    - Use concurrent requests where possible to speed up processing.
2. **Scalability**:

    - The system should support expanding the extraction rules for different data fields in the future.

3. **Reliability**:
    - Implement retry mechanisms for failed requests.
4. **Usability**:
    - Document how to run the scripts and set up the folder structure.
5. **Maintainability**:
    - Code should be modular, with separate functions for loading data, scraping, parsing, and saving results.

4. System Constraints

1. The input CSV file should have a column containing URLs in a consistent format.
2. An internet connection is required for scraping.
3. Some URLs may be blocked or rate-limited, requiring delays or proxy use.
4. Python 3.x must be installed along with dependencies listed in `requirements.txt`.

5. Assumptions

1. Claim dates appear in a recognisable and consistent HTML pattern for most URLs.
2. The server hosting the URLs allows scraping within reasonable limits.
3. The user has permissions to process and store the extracted data.