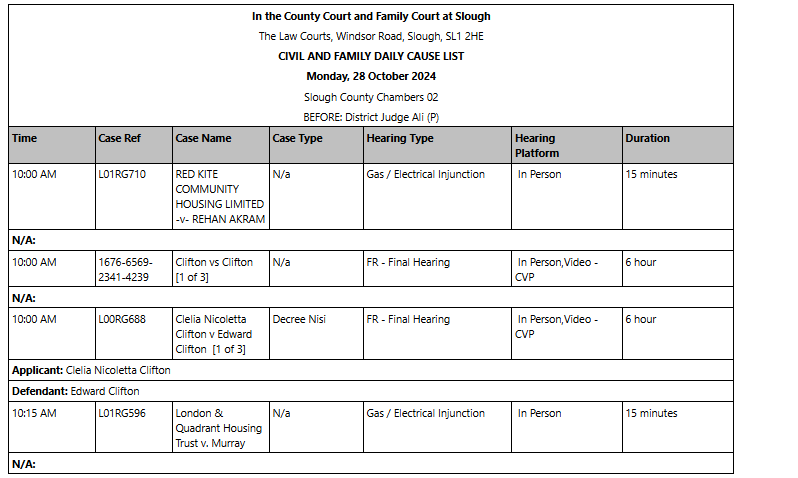
### **Documentation for the HTML Template Data Identification and Extraction Script**

This Node.js script allows for the identification and extraction of data from various specific HTML templates often used in legal contexts, organizing it into a tabular format for analysis. The script's logic is based on identifying textual and structural patterns within the HTML files, utilizing the cheerio library for DOM manipulation and parsing.

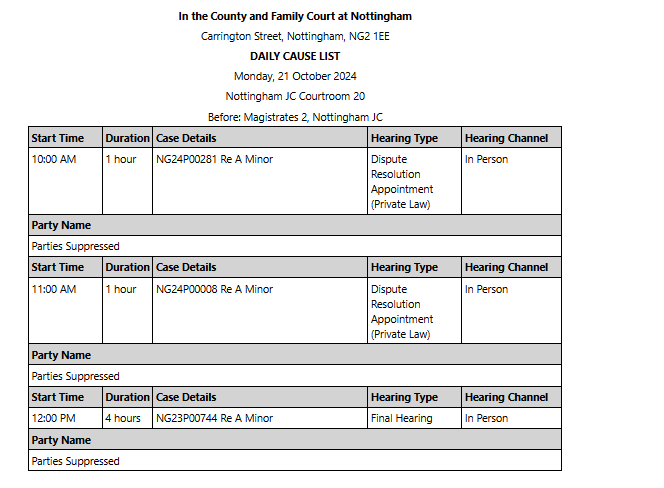
### **Overview of Functionality**

1. **Loading and Normalizing HTML**:
   * The HTML content is loaded and converted into a manipulatable object using cheerio.
   * The text of each template is normalized (converted to lowercase and excess spaces removed) to standardize the analysis.
2. **Template Identification**:
   * The identifyTemplate function examines the HTML content to determine which of the templates it corresponds to. The function looks for keywords and specific headers, such as:

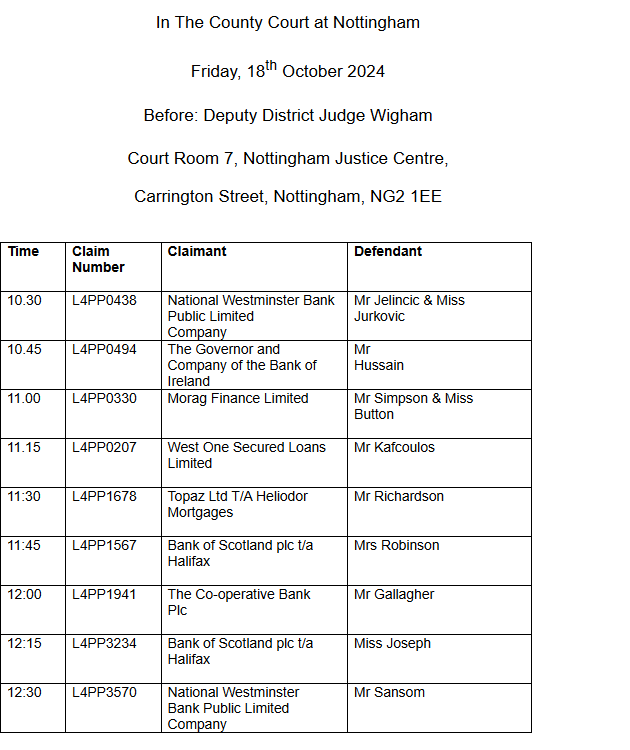
**Template7**: Contains headers like "Case Ref" and "Case Name."



**Template5**: Includes headers like "Start Time," "Duration," "Case Details," among others.



* **Template4 and Template4a**: Contain variations of "Claim Number", "Claimant," and "Defendant."



* + Each template follows a specific formatting and content pattern, allowing the function to return a string identifying the template.

1. **Data Extraction by Template**:
   * **Template4**: Extracts data such as "Claim Number," "Claimant," and "Defendant" from structured tables without a combined header.
   * **Template4a**: Similar to Template4 but with a combined header like "Claim Number Claimant."
   * **Template5**: Extracts hearing information such as "Start Time," "Duration," "Case Details," "Hearing Type," and "Hearing Channel," while identifying the claimant and defendant in the "Case Details" field.
   * **Template7**: Identifies hearings based on case references like "Case Ref."
2. **Data Structuring**:
   * For each extracted data row, an object structure is created with relevant fields such as "Court Name," "Court Date," "Claim Number," "Claimant," "Defendant," "Duration," among others.
   * Each object is added to a list representing a complete row of data extracted from the HTML table.
3. **CSV Generation**:
   * After data extraction, the script stores the information in a CSV file.
   * The CSV format contains fixed columns for each template, including:

**Court Name**: Name of the court.

**Court Date**: Date of the hearing or court session.

**Claim Number**: Case number or reference.

**Claimant**: Name of the claimant.

**Defendant**: Name of the defendant.

**Duration**: Duration of the hearing (if available).

**Hearing Type**: Type of hearing (if available).

**Hearing Channel**: Channel of the hearing (if available).

### 

### 

### 

### 

### 

### 

### 

### **Example CSV Structure**

The generated CSV will have a structure similar to:

| **Court Name** | **Court Date** | **Claim Number** | **Claimant** | **Defendant** | **Duration** | **Hearing Type** | **Hearing Channel** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Tribunal XYZ | 2023-10-27 | CN1234 | John Doe | Jane Roe | 2 hours | Online | Zoom |
| Tribunal ABC | 2023-11-03 | CN5678 | Alice Smith | Bob Brown | 1.5 hours | Physical | Courtroom A |
| ... | ... | ... | ... | ... | ... | ... | ... |

### 

### 

### 

### **Conclusion**

This script is useful for automating the process of extracting data from structured HTML files, especially in contexts where multiple template formats are used. The final result is a consolidated CSV with essential hearing data, facilitating analysis and report generation for the user.