Project Report: Web Scraping of Court Judgments

Introduction

This project is a web scraping tool designed to automate the process of retrieving court judgment data from an online platform. The system extracts details such as case type, case number, diary year, and CAPTCHA resolution, and saves the data into a CSV file categorized by High Courts.

Tools and Libraries Used

1. Selenium

- **Purpose**: To automate browser interactions such as filling forms, selecting dropdown values, and clicking buttons.
- **Functions Used**:
    - webdriver.Chrome(): Initializes the Chrome WebDriver.
    - WebDriverWait: Waits dynamically for elements to load.
    - Select: Interacts with dropdown menus.
    - find_element, find_elements: Locates elements on the webpage.

2. pandas

- **Purpose**: To organize and store scraped data into a structured CSV format.
- **Functions Used**:
    - pandas.DataFrame: Creates a data frame from the extracted data.
    - to_csv: Saves the data frame to a CSV file.

3. Python's Built-in Libraries

- **time**: Introduces delays for content to load dynamically.
- **input**: Manages manual entry for CAPTCHA resolution.

4. Browser Drivers

- **ChromeDriver**: A driver to interact with Google Chrome for running Selenium-based automation.

Steps Involved

1. Environment Setup

- Installed necessary libraries using pip:
- pip install selenium pandas

- Installed and configured ChromeDriver to ensure compatibility with the Chrome browser.

2. Building Modules

a. scraper.py

- **Purpose**: Automates the web scraping workflow.
- **Steps**:
    1. **Navigate to the URL**:
        - Open the target webpage using Selenium's webdriver.
    2. **Select Case Type and Diary Year**:
        - Dropdown menus were handled using Selenium's Select.
    3. **Enter Case Number**:
        - Case number is entered in the designated input field using send_keys.
    4. **Solve CAPTCHA**:
        - CAPTCHA text is extracted and manually solved or handled via captcha_solver.py.
        - The answer is entered and submitted.
    5. **Scrape Case Details**:
        - Extract case titles, dates, and summaries using find_elements.
        - Save the scraped data into cases.csv for later analysis.
    6. **Handle Pagination**:
        - Dynamically navigate through pages using the "Next" button until all data is scraped.

b. captcha_solver.py

- **Purpose**: Provides an interface to solve CAPTCHA challenges. It currently uses manual input but can be extended with services like DeathByCaptcha or reCAPTCHA solvers.

c. setup.py

- **Purpose**: Defines dependencies and installation requirements for the project.

3. Testing and Debugging

- Verified element locators using browser developer tools (Inspect Element) to extract IDs, classes, and other attributes.
- Ensured dynamic elements were handled using WebDriverWait.
- Fixed errors such as missing configurations (config.py) and incorrect element locators.

4. Output Handling

- Organized scraped data into a CSV file (cases.csv) inside the data/ directory.
- Verified the output for consistency and correctness.

Challenges Faced

1. **Handling CAPTCHA**:
   - CAPTCHA required manual solving, delaying the process. Automation using third-party services can resolve this issue.
2. **Dynamic Page Loading**:
   - Implemented WebDriverWait to handle slow-loading elements effectively.
3. **Time Constraints**:
   - Limited focus on the project due to overlapping exams.

Conclusion

All tools, libraries, and code required for this project are ready. However, the web scraping process requires considerable time and attention to detail, especially with manual CAPTCHA handling and navigating dynamic elements.

Future Work

Given more time, the following improvements will be made:

1. Integrate an automated CAPTCHA solver to streamline the workflow.
2. Optimize the scraping script for faster execution and error handling.
3. Add additional features, such as scraping other data categories.

Final Note

This project was interrupted by exams, limiting my ability to focus fully. If additional time is granted, I will surely improve and complete this project to its full potential.