KYC WEB SCRAPING PROJECT

1. Introduction

The Know Your Customer (KYC) Web Scraping Project focuses on automating the extraction of public data related to legislators and government officials. The goal is to collect structured information from web pages efficiently and accurately using web scraping techniques.

This project leverages **Playwright**, **Jsoup**, and **JSON handling** to gather data, clean it, and convert it into a usable structured format. It forms part of a larger effort to integrate automation into information-gathering workflows.

2. Objective

The main objective is to design and implement a web scraper that can extract tabular data, process it, and store it in a readable format like JSON. The project aims to demonstrate:

- Automation in web-based data extraction
- Parsing HTML using Java tools
- Saving data in standardized, machine-readable formats

3. Tools and Technologies Used

- **Programming Language:** Java
- **Automation Library:** Microsoft Playwright
- **HTML Parser:** Jsoup
- **Data Format:** JSON
- **Build Tool:** Gradle
- **IDE:** VS Code

4. Methodology

1. **Website Selection:**
   The chosen website is the Alaska State Legislature official site — https://akleg.gov/senate.php.

2. **Data Extraction:**
   The scraper navigates the page, reads HTML tables containing senator details, and extracts name, district, and party information.

3. **HTML Parsing:**
   Jsoup is used to parse the fetched HTML content and identify relevant data fields.

4. **Data Storage:**
   The data is converted into JSON format and written to a file (`senators.json`).

5. **Automation:**
   Playwright ensures smooth page loading and reliability in data fetching.

5. System Design

The project is structured in a modular way:

- **Scraper.java** — contains Playwright and Jsoup integration logic.
- **Data Model:** JSON object schema representing each senator.
- **Output File:** Automatically created and overwritten when executed again.

Flow:

```
URL → Playwright (fetch HTML) → Jsoup (parse data) → JSON → File output
```

6. Key Features

- Headless browser automation with Playwright
- Dynamic content extraction
- Error handling for missing or invalid data
- Automatic file overwrite for repeated runs
- Structured JSON output

7. Challenges Faced

- Handling timeouts due to slow page loads
- Ensuring compatibility across different systems
- Parsing inconsistencies caused by minor website structure changes
- Managing dependencies and version mismatches in Maven

8. Real-World Applications

- **Data Aggregation:** Gathering information from official or public databases.
- **Market Research:** Extracting structured insights from e-commerce or public sources.
- **Government Transparency Tools:** Automating data collection for open data initiatives.
- **KYC Verification:** Collecting and cross-referencing entity data for compliance workflows.

9. Future Scope

- Extending the scraper to cover multiple legislative chambers or countries.
- Integrating AI-based data validation and entity matching.
- Building a user-friendly dashboard to visualize collected information.
- Adding scheduling for periodic automated updates.

10. Time Taken

- **Research & Setup:** 2 hours
- **Coding & Debugging:** 3 hours
- **Testing & Documentation:** 1 hour
  Total Time: 6 hours

11. Conclusion & Learnings

Through this project, the importance of automation in web data extraction became evident. The experience strengthened skills in Java, Playwright automation, and HTML parsing.

This system can serve as a foundation for advanced data collection pipelines or compliance-oriented applications. It demonstrates how clean, structured data can be efficiently derived from unstructured online sources.

Submitted by:

Name: Nikita
Email: nikita.nik393@gmail.com
Enrollment No: 12614803122